```
                        title   'CP/M BDOS Interface, BDOS, Version 3.0 Dec, 1982'
                ;*************************************************************
                ;*************************************************************
                ;**                                                         **
                ;**  B a s i c   D i s k   O p e r a t i n g   S y s t e m  **
                ;**                                                         **
                ;**            I n t e r f a c e   M o d u l e              **
                ;**                                                         **
                ;*************************************************************
                ;*************************************************************
                ;
                ;       Copyright (c) 1978, 1979, 1980, 1981, 1982
                ;       Digital Research
                ;       Box 579, Pacific Grove
                ;       California
                ;
                ;        December 1982
                ;
 FFFF =         on      equ     0ffffh
 0000 =         off     equ     00000h
 0000 =         MPM     equ     off
 0000 =         BANKED  equ     off

                ;
                ;       equates for non graphic characters
                ;

 0001 =         ctla    equ     01h     ; control a
 0002 =         ctlb    equ     02h     ; control b
 0003 =         ctlc    equ     03h     ; control c
 0005 =         ctle    equ     05h     ; physical eol
 0006 =         ctlf    equ     06h     ; control f
 0007 =         ctlg    equ     07h     ; control g
 0008 =         ctlh    equ     08h     ; backspace
 000B =         ctlk    equ     0bh     ; control k
 0010 =         ctlp    equ     10h     ; prnt toggle
 0011 =         ctlq    equ     11h     ; start screen
 0012 =         ctlr    equ     12h     ; repeat line
 0013 =         ctls    equ     13h     ; stop screen
 0015 =         ctlu    equ     15h     ; line delete
 0017 =         ctlw    equ     17h     ; control w
 0018 =         ctlx    equ     18h     ; =ctl-u
 001A =         ctlz    equ     1ah     ; end of file
 007F =         rubout  equ     7fh     ; char delete
 0009 =         tab     equ     09h     ; tab char
 000D =         cr      equ     0dh     ; carriage return
 000A =         lf      equ     0ah     ; line feed
 005E =         ctl     equ     5eh     ; up arrow

 0000                           org     0000h
 0000 =         base            equ     $

                ; Base page definitions

 FC00 =         bnkbdos$pg      equ     base+0fc00h
 FD00 =         resbdos$pg      equ     base+0fd00h
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
FB00 =          scb$pg          equ     base+0fb00h
FF00 =          bios$pg                 equ     base+0ff00h

                ; Bios equates

FF00 =          bios            equ     bios$pg
FF00 =          bootf           equ     bios$pg         ; 00. cold boot function

                if BANKED

                wbootf          equ     scb$pg+68h      ; 01. warm boot function
                constf          equ     scb$pg+6eh      ; 02. console status function
                coninf          equ     scb$pg+74h      ; 03. console input function
                conoutf                 equ     scb$pg+7ah      ; 04. console output function
                listf           equ     scb$pg+80h      ; 05. list output function

                else

FF03 =          wbootf          equ     bios$pg+3       ; 01. warm boot function
FF06 =          constf          equ     bios$pg+6       ; 02. console status function
FF09 =          coninf          equ     bios$pg+9       ; 03. console input function
FF0C =          conoutf                 equ     bios$pg+12      ; 04. console output function
FF0F =          listf           equ     bios$pg+15      ; 05. list output function

                endif

FF12 =          punchf          equ     bios$pg+18      ; 06. punch output function
FF15 =          readerf                 equ     bios$pg+21      ; 07. reader input function
FF18 =          homef           equ     bios$pg+24      ; 08. disk home function
FF1B =          seldskf                 equ     bios$pg+27      ; 09. select disk function
FF1E =          settrkf                 equ     bios$pg+30      ; 10. set track function
FF21 =          setsecf                 equ     bios$pg+33      ; 11. set sector function
FF24 =          setdmaf                 equ     bios$pg+36      ; 12. set dma function
FF27 =          readf           equ     bios$pg+39      ; 13. read disk function
FF2A =          writef          equ     bios$pg+42      ; 14. write disk function
FF2D =          liststf                 equ     bios$pg+45      ; 15. list status function
FF30 =          sectran                 equ     bios$pg+48      ; 16. sector translate
FF33 =          conoutstf       equ     bios$pg+51      ; 17. console output status function
FF36 =          auxinstf        equ     bios$pg+54      ; 18. aux input status function
FF39 =          auxoutstf       equ     bios$pg+57      ; 19. aux output status function
FF3C =          devtblf                 equ     bios$pg+60      ; 20. retunr device table address fx
FF3F =          devinitf        equ     bios$pg+63      ; 21. initialize device function
FF42 =          drvtblf                 equ     bios$pg+66      ; 22. return drive table address
FF45 =          multiof                 equ     bios$pg+69      ; 23. multiple i/o function
FF48 =          flushf          equ     bios$pg+72      ; 24. flush function
FF4B =          movef           equ     bios$pg+75      ; 25. memory move function
FF4E =          timef           equ     bios$pg+78      ; 26. system get/set time function
FF51 =          selmemf                 equ     bios$pg+81      ; 27. select memory function
FF54 =          setbnkf                 equ     bios$pg+84      ; 28. set dma bank function
FF57 =          xmovef          equ     bios$pg+87      ; 29. extended move function

                if BANKED

                ; System Control Block equates

                olog            equ     scb$pg+090h
```

```
rlog            equ     scb$pg+092h

SCB             equu    scb$pg+09ch

; Expansion Area - 6 bytes

hashl           equ     scb$pg+09ch
hash            equ     scb$pg+09dh
version                 equ     scb$pg+0a1h

; Utilities Section - 8 bytes

util$flgs       equ     scb$pg+0a2h
dspl$flgs       equ     scb$pg+0a6h

; CLP Section - 4 bytes

clp$flgs        equ     scb$pg+0aah
clp$errcde      equ     scb$pg+0ach

; CCP Section - 8 bytes

ccp$comlen      equ     scb$pg+0aeh
ccp$curdrv      equ     scb$pg+0afh
ccp$curusr      equ     scb$pg+0b0h
ccp$conbuff     equ     scb$pg+0b1h
ccp$flgs        equ     scb$pg+0b3h

; Device I/O Section - 32 bytes

conwidth        equ     scb$pg+0b6h
column          equ     scb$pg+0b7h
conpage                 equ     scb$pg+0b8h
conline                 equ     scb$pg+0b9h
conbuffadd      equ     scb$pg+0bah
conbufflen      equ     scb$pg+0bch
conin$rflg      equ     scb$pg+0beh
conout$rflg     equ     scb$pg+0c0h
auxin$rflg      equ     scb$pg+0c2h
auxout$rflg     equ     scb$pg+0c4h
lstout$rflg     equ     scb$pg+0c6h
page$mode       equ     scb$pg+0c8h
pm$default      equ     scb$pg+0c9h
ctlh$act        equ     scb$pg+0cah
rubout$act      equ     scb$pg+0cbh
type$ahead      equ     scb$pg+0cch
contran                 equ     scb$pg+0cdh
conmode                 equ     scb$pg+0cfh
outdelim        equ     scb$pg+0d3h
listcp          equ     scb$pg+0d4h
qflag           equ     scb$pg+0d5h

; BDOS Section - 42 bytes

scbadd          equ     scb$pg+0d6h
dmaad           equ     scb$pg+0d8h
```

```
                olddsk          equ     scb$pg+0dah
                info            equ     scb$pg+0dbh
                resel           equ     scb$pg+0ddh
                relog           equ     scb$pg+0deh
                fx              equ     scb$pg+0dfh
                usrcode                 equ     scb$pg+0e0h
                dcnt            equ     scb$pg+0e1h
                ;searcha        equ     scb$pg+0e3h
                searchl                 equ     scb$pg+0e5h
                multcnt                 equ     scb$pg+0e6h
                errormode       equ     scb$pg+0e7h
                searchchain     equ     scb$pg+0e8h
                temp$drive      equ     scb$pg+0ech
                errdrv          equ     scb$pg+0edh
                media$flag      equ     scb$pg+0f0h
                bdos$flags      equ     scb$pg+0f3h
                stamp           equ     scb$pg+0f4h
                commonbase      equ     scb$pg+0f9h
                error           equ     scb$pg+0fbh     ;jmp error$sub
                bdosadd                 equ     scb$pg+0feh

                ; Resbdos equates

                resbdos                 equ     resbdos$pg
                move$out        equ     resbdos$pg+9    ; a=bank #, hl=dest, de=srce
                move$tpa        equ     resbdos$pg+0ch  ; a=bank #, hl=dest, de=srce
                srch$hash       equ     resbdos$pg+0fh  ; a=bank #, hl=hash table addr
                hashmx          equ     resbdos$pg+12h  ; max hash search dcnt
                rd$dir$flag     equ     resbdos$pg+14h  ; directory read flag
                make$xfcb       equ     resbdos$pg+15h  ; make function flag
                find$xfcb       equ     resbdos$pg+16h  ; search function flag
                xdcnt           equ     resbdos$pg+17h  ; dcnt save for empty fcb,
                                                        ; user 0 fcb, or xfcb
                xdmaad          equ     resbdos$pg+19h  ; resbdos dma copy area addr
                curdma          equ     resbdos$pg+1bh  ; current dma
                copy$cr$only    equ     resbdos$pg+1dh  ; dont restore fcb flag
                user$info       equ     resbdos$pg+1eh  ; user fcb address
                kbchar          equ     resbdos$pg+20h  ; conbdos look ahead char
                qconinx                 equ     resbdos$pg+21h  ; qconin mov a,m routine

                ELSE

FF4B =          move$out        equ     movef
FF4B =          move$tpa        equ     movef

                ENDIF

                ;
0000 3635343332serial: db       '654321'
                ;
                ;       Enter here from the user's program with function number in c,
                ;       and information address in d,e
                ;

                bdose: ; Arrive here from user programs
0006 EB22DB1EEB         xchg! shld info! xchg ; info=de, de=info
```

```

000B 7932DF1EFE         mov a,c! sta fx! cpi 14! jc bdose2
0014 21000022BD         lxi h,0! shld dircnt ; dircnt,multnum = 0
001A 3ADA1E32C6         lda olddsk! sta seldsk ; Set seldsk

                if BANKED
                        dcr a! sta copy$cr$init
                ENDIF

                        ; If mult$cnt ~= 1 then read or write commands
                        ; are handled by the shell
0020 3AE61E3DCA         lda mult$cnt! dcr a! jz bdose2
0027 219400             lxi h,mult$fxs
                bdose1:
002A 7EB7CA3700         mov a,m! ora a! jz bdose2
002F B9CACD02           cmp c! jz shell
0033 23C32A00           inx h! jmp bdose1
                bdose2:
0037 7B32C31C           mov a,e! sta linfo ; linfo = low(info) - don't equ
003B 210000227A         lxi h,0! shld aret ; Return value defaults to 0000
0041 22DD1E             shld resel ; resel,relog = 0
                        ; Save user's stack pointer, set to local stack
0044 3922CB02           dad sp! shld entsp ; entsp = stackptr

                if not BANKED
0048 31EC00             lxi sp,lstack ; local stack setup
                ENDIF

004B 21641C             lxi h,goback ; Return here after all functions
004E E5                 push h ; jmp goback equivalent to ret
004F 79FE33D25C         mov a,c! cpi nfuncs! jnc high$fxs ; Skip if invalid #
0055 4B                 mov c,e ; possible output character to c
0056 21EC00C36E         lxi h,functab! jmp bdos$jmp
                        ; look for functions 98 ->
                high$fxs:
005C FE80D28201         cpi 128! jnc test$152
0061 D662DA260F         sui 98! jc lret$eq$ff ; Skip if function < 98
0066 FE0FD2260F         cpi nfuncs2! jnc lret$eq$ff
006B 215201             lxi h,functab2
                bdos$jmp:
006E 5F1600             mov e,a! mvi d,0 ; de=func, hl=.ciotab
0071 19195E2356         dad d! dad d! mov e,m! inx h! mov d,m ; de=functab(func)
0076 2ADB1E             lhld info ; info in de for later xchg
0079 EBE9               xchg! pchl ; dispatched

                ;          CAUTION: In banked systems only,
                ;           error$sub is referenced indirectly by the SCB ERROR
                ;          field in RESBDOS as (0fc7ch).  This value is converted
                ;          to the actual address of error$sub by GENSYS.  If the offset
                ;          of error$sub is changed, the SCB ERROR value must also
                ;          be changed.

                ;
                ;       error subroutine
                ;

```

```
                     error$sub:
  007B 0600C50D             mvi b,0! push b! dcr c
  007F 2170010909           lxi h,errtbl! dad b! dad b
  0084 5E2356EB             mov e,m! inx h! mov d,m! xchg
  0088 CDA602               call errflg
  008B C13AE71EB7           pop b! lda error$mode! ora a! rnz
  0091 C3BC02               jmp reboote

  0094 1415212228mult$fxs:          db      20,21,33,34,40,0

                     if BANKED
                            db      'COPYRIGHT (C) 1982,'
                            db      ' DIGITAL RESEARCH '
                            db      '151282'
                     else
  009A 434F50522E           db      'COPR. ''82 DRI 151282'

                            ;       31 level stack

  00AE C7C7C7C7C7           dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
  00BC C7C7C7C7C7           dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
  00CC C7C7C7C7C7           dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
  00DC C7C7C7C7C7           dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
                     lstack:

                     endif

                     ;      dispatch table for functions

                     functab:
  00EC C8025E03A0           dw      rebootx1, func1, func2, func3
  00F4 12FF0FFF6A           dw      punchf, listf, func6, func7
  00FC 94069A061D           dw      func8, func9, func10, func11
  000C =             diskf  equ     ($-functab)/2   ; disk funcs
  0104 121818182B           dw      func12,func13,func14,func15
  010C B818CC181D           dw      func16,func17,func18,func19
  0114 2E19371940           dw      func20,func21,func22,func23
  011C 8F1995199B           dw      func24,func25,func26,func27
  0124 910BAB19B1           dw      func28,func29,func30,func31
  012C C719DB19E4           dw      func32,func33,func34,func35
  0134 5516F319CE           dw      func36,func37,func38,func39
  013C E419260FCE           dw      func40,lret$eq$ff,func42,func43
  0144 171A251A2A           dw      func44,func45,func46,func47
  014C 791AC41AEA           dw      func48,func49,func50
  0033 =             nfuncs equ     ($-functab)/2

                     functab2:
  0152 821A161B             dw      func98,func99
  0156 260FEE1BF7           dw      func100,func101,func102,func103
  015E 251C321CCE           dw      func104,func105,func106,func107
  0166 501CD406E2           dw      func108,func109,func110,func111
  016E EE06                 dw      func112

  000F =             nfuncs2        equ     ($-functab2)/2

                     errtbl:
```

```
0170 2B07               dw      permsg
0172 5107               dw      rodmsg
0174 4207               dw      rofmsg
0176 3407               dw      selmsg
0178 0000               dw      0
017A 0000               dw      0
017C 6007               dw      passmsg
017E 6007               dw      fxstsmsg
0180 6C07               dw      wildmsg

                test$152:
0182 FE98C0             cpi 152! rnz

                ;
                ;       PARSE version 3.0b  Oct 08 1982 - Doug Huskey
                ;
                ;

                        ; DE->.(.filename,.fcb)
                        ;
                        ; filename = [d:]file[.type][;password]
                        ;
                        ; fcb assignments
                        ;
                        ;   0     => drive, 0 = default, 1 = A, 2 = B, ...
                        ;   1-8   => file, converted to upper case,
                        ;            padded with blanks (left justified)
                        ;   9-11  => type, converted to upper case,
                        ;            padded with blanks (left justified)
                        ;   12-15 => set to zero
                        ;   16-23 => password, converted to upper case,
                        ;            padded with blanks
                        ;   24-25 => 0000h
                        ;   26    => length of password (0 - 8)
                        ;
                        ; Upon return, HL is set to FFFFH if DE locates
                        ;            an invalid file name;
                        ; otherwise, HL is set to 0000H if the delimiter
                        ;            following the file name is a 00H (NULL)
                        ;            or a 0DH (CR);
                        ; otherwise, HL is set to the address of the delimiter
                        ;            following the file name.
                        ;
0185 21C319             lxi     h,sthl$ret
0188 E5                 push    h
0189 2ADB1E             lhld    info
018C 5E                 mov     e,m             ;get first parameter
018D 23                 inx     h
018E 56                 mov     d,m
018F D5                 push    d               ;save .filename
0190 23                 inx     h
0191 5E                 mov     e,m             ;get second parameter
0192 23                 inx     h
0193 56                 mov     d,m
0194 E1                 pop     h               ;DE=.fcb  HL=.filename
0195 EB                 xchg
                parse0:
```

```
0196 E5                 push    h               ;save .fcb
0197 AF                 xra     a
0198 77                 mov     m,a             ;clear drive byte
0199 23                 inx     h
019A 010B20             lxi     b,20h*256+11
019D CD9202             call    pad             ;pad name and type w/ blanks
01A0 010400             lxi     b,4
01A3 CD9202             call    pad             ;EXT, S1, S2, RC = 0
01A6 010820             lxi     b,20h*256+8
01A9 CD9202             call    pad             ;pad password field w/ blanks
01AC 010C00             lxi     b,12
01AF CD9202             call    pad             ;zero 2nd 1/2 of map, cr, r0 - r2
                ;
                ;       skip spaces
                ;
01B2 CD9902             call    skps
                ;
                ;       check for drive
                ;
01B5 1A                 ldax    d
01B6 FE3A               cpi     ':'             ;is this a drive?
01B8 1B                 dcx     d
01B9 E1                 pop     h
01BA E5                 push    h               ;HL = .fcb
01BB C2D201             jnz     parse$name
                ;
                ;       Parse the drive-spec
                ;
                parsedrv:
01BE CD7302             call    delim
01C1 CA0602             jz      parse$ok
01C4 D641               sui     'A'
01C6 DA1C02             jc      perror1
01C9 FE10               cpi     16
01CB D21C02             jnc     perror1
01CE 13                 inx     d
01CF 13                 inx     d               ;past the ':'
01D0 3C                 inr     a               ;set drive relative to 1
01D1 77                 mov     m,a             ;store the drive in FCB(0)
                ;
                ;       Parse the file-name
                ;
                parse$name:
01D2 23                 inx     h               ;HL = .fcb(1)
01D3 CD7302             call    delim
01D6 CA0602             jz      parse$ok
01D9 010007             lxi     b,7*256

01DC 1A         parse6:         ldax    d               ;get a character
01DD FE2E               cpi     '.'             ;file-type next?
01DF CAF001             jz      parse$type      ;branch to file-type processing
01E2 FE3B               cpi     ';'
01E4 CA2102             jz      parse$pw
01E7 CD3F02             call    gfc             ;process one character
01EA C2DC01             jnz     parse6          ;loop if not end of name
01ED C30602             jmp     parse$ok
```

```
                        ;
                        ;       Parse the file-type
                        ;
                        parse$type:
01F0 13                         inx     d               ;advance past dot
01F1 E1                         pop     h
01F2 E5                         push    h               ;HL =.fcb
01F3 010900                     lxi     b,9
01F6 09                         dad     b               ;HL =.fcb(9)
01F7 010002                     lxi     b,2*256

01FA 1A                 parse8:         ldax    d
01FB FE3B                       cpi     ';'
01FD CA2102                     jz      parsepw
0200 CD3F02                     call    gfc             ;process one character
0203 C2FA01                     jnz     parse8          ;loop if not end of type
                        ;
                        parse$ok:
0206 C1                         pop     b
0207 D5                         push    d
0208 CD9902                     call    skps            ;skip trailing blanks and tabs
020B 1B                         dcx     d
020C CD7302                     call    delim           ;is next nonblank char a delim?
020F E1                         pop     h
0210 C0                         rnz                     ;no
0211 210000                     lxi     h,0
0214 B7                         ora     a
0215 C8                         rz                      ;return zero if delim = 0
0216 FE0D                       cpi     cr
0218 C8                         rz                      ;return zero if delim = cr
0219 EB                         xchg
021A C9                         ret
                        ;
                        ;       handle parser error
                        ;
                        perror:
021B C1                         pop     b                       ;throw away return addr
                        perror1:
021C C1                         pop     b
021D 21FFFF                     lxi     h,0ffffh
0220 C9                         ret
                        ;
                        ;       Parse the password
                        ;
                        parsepw:
0221 13                         inx     d
0222 E1                         pop     h
0223 E5                         push    h
0224 011000                     lxi     b,16
0227 09                         dad     b
0228 010107                     lxi     b,7*256+1
                        parsepw1:
022B CD3F02                     call    gfc
022E C22B02                     jnz     parsepw1
0231 3E07                       mvi     a,7
0233 90                         sub     b
```

```
0234 E1                  pop     h
0235 E5                  push    h
0236 011A00              lxi     b,26
0239 09                  dad     b
023A 77                  mov     m,a
023B 1A                  ldax    d                       ;delimiter in A
023C C30602              jmp     parse$ok
               ;
               ;         get next character of name, type or password
               ;
023F CD7302    gfc:      call    delim           ;check for end of filename
0242 C8                  rz                      ;return if so
0243 FE20                cpi     ' '             ;check for control characters
0245 13                  inx     d
0246 DA1B02              jc      perror          ;error if control characters encountered
0249 04                  inr     b               ;error if too big for field
024A 05                  dcr     b
024B FA1B02              jm      perror
024E 0C                  inr     c
024F 0D                  dcr     c
0250 C25802              jnz     gfc1
0253 FE2A                cpi     '*'             ;trap "match rest of field" character
0255 CA5D02              jz      setmatch
0258 77        gfc1:     mov     m,a             ;put character in fcb
0259 23                  inx     h
025A 05                  dcr     b               ;decrement field size counter
025B B7                  ora     a               ;clear zero flag
025C C9                  ret
               ;;
               setmatch:
025D 363F                mvi     m,'?'           ;set match one character
025F 23                  inx     h
0260 05                  dcr     b
0261 F25D02              jp      setmatch
0264 C9                  ret
               ;
               ;         check for delimiter
               ;
               ;         entry:  A = character
               ;         exit:   z = set if char is a delimiter
               ;
0265 0D09202E2Cdelimiters:    db      cr,tab,' .,:;[]=<>|',0

0273 1A        delim: ldax    d                  ;get character
0274 E5                  push    h
0275 216502              lxi     h,delimiters
0278 BE        delim1:        cmp     m                  ;is char in table
0279 CA8302              jz      delim2
027C 34                  inr     m
027D 35                  dcr     m               ;end of table? (0)
027E 23                  inx     h
027F C27802              jnz     delim1
0282 B7                  ora     a               ;reset zero flag
0283 E1        delim2:        pop     h
0284 C8                  rz
                         ;
```

```
                        ;       not a delimiter, convert to upper case
                        ;
0285 FE61               cpi     'a'
0287 D8                 rc
0288 FE7B               cpi     'z'+1
028A D28F02             jnc     delim3
028D E65F               ani     05fh
028F E67F       delim3:         ani     07fh
0291 C9                 ret                     ;return with zero set if so
                ;
                ;       pad with blanks or zeros
                ;
0292 70         pad:    mov     m,b
0293 23                 inx     h
0294 0D                 dcr     c
0295 C29202             jnz     pad
0298 C9                 ret
                ;
                ;       skip blanks and tabs
                ;
0299 1A         skps:   ldax    d
029A 13                 inx     d
029B FE20               cpi     ' '             ;skip spaces & tabs
029D CA9902             jz      skps
02A0 FE09               cpi     tab
02A2 CA9902             jz      skps
02A5 C9                 ret
                ;
                ;       end of PARSE
                ;

                errflg:
                        ; report error to console, message address in hl
02A6 E5CDE804           push h! call crlf ; stack mssg address, new line
02AA 3ACC1CC641         lda adrive! adi 'A'! sta dskerr ; current disk name
02B2 011907             lxi b,dskmsg

                if BANKED
                        call zprint ; the error message
                else
02B5 CDF204             call print
                endif

02B8 C1                 pop b

                if BANKED
                        lda bdos$flags! ral! jnc zprint
                        call zprint ; error message tail
                        lda fx! mvi b,30h
                        lxi h,pr$fx1
                        cpi 100! jc errflg1
                        mvi m,31h! inx h! sui 100
                errflg1:
                        sui 10! jc errflg2
                        inr b! jmp errflg1
                errflg2:
```

```
                          mov m,b! inx h! adi 3ah! mov m,a
                          inx h! mvi m,20h
                          lxi h,pr$fcb! mvi m,0
                          lda resel! ora a! jz errflg3
                          mvi m,20h! push d
                          lhld info! inx h! xchg! lxi h,pr$fcb1
                          mvi c,8! call move! mvi m,'.'! inx h
                          mvi c,3! call move! pop d
                  errflg3:
                          call crlf
                          lxi b,pr$fx! jmp zprint

                  zprint:
                          ldax b! ora a! rz
                          push b! mov c,a
                          call tabout
                          pop b! inx b! jmp zprint

                  pr$fx: db       'BDOS Function = '
                  pr$fx1:         db      '   '
                  pr$fcb:         db      ' File = '
                  pr$fcb1:ds      12
                          db      0

                  else
02B9 C3F204               jmp     print
                  endif

                  reboote:
02BC 21FDFFC3C5           lxi h,0fffdh! jmp rebootx0 ; BDOS error
                  rebootx:
02C2 21FEFF               lxi h,0fffeh ; CTL-C error
                  rebootx0:
02C5 22AC1E               shld clp$errcde
                  rebootx1:
02C8 C303FF               jmp wbootf

02CB              entsp: ds       2       ; entry stack pointer

                  shell:
02CD 2100003922           lxi h,0! dad sp! shld shell$sp

                  if not BANKED
02D4 310E03               lxi sp,shell$stk
                  endif

02D7 213E03E5             lxi h,shell$rtn! push h
02DB CD0507CD25           call save$rr! call save$dma
02E1 3AE61E               lda mult$cnt
                  mult$io:
02E4 F532BE1CCD           push a! sta mult$num! call cbdos
02EB B7C23503             ora a! jnz shell$err
02EF 3ADF1EFE21           lda fx! cpi 33! cnc incr$rr
02F7 CD1B03               call adv$dma
02FA F13DC2E402           pop a! dcr a! jnz mult$io
02FF 676FC9               mov h,a! mov l,a! ret
```

```

0302 0000      shell$sp:     dw     0

0304 C7C7C7C7C7              dw     0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h

               shell$stk:           ; shell has 5 level stack
030E 0000      hold$dma:     dw     0

               cbdos:
0310 3ADF1E4F         lda fx! mov c,a
               cbdos1:
0314 2ADB1EEBC3       lhld info! xchg! jmp bdose2

               adv$dma:
031B 2AD81E1180       lhld dmaad! lxi d,80h! dad d! jmp reset$dma1

               save$dma:
0325 2AD81E220E       lhld dmaad! shld hold$dma! ret

               reset$dma:
032C 2A0E03           lhld hold$dma
               reset$dma1:
032F 22D81EC35B       shld dmaad! jmp setdma

               shell$err:
0335 C13CC8           pop b! inr a! rz
0338 3AE61E9067       lda mult$cnt! sub b! mov h,a! ret

               shell$rtn:
033E E53ADF1EFE       push h! lda fx! cpi 33! cnc reset$rr
0347 CD2C03           call reset$dma
034A D12A0203F9       pop d! lhld shell$sp! sphl! xchg
0350 7D44C9           mov a,l! mov b,h! ret

                      page



                      title   'CP/M Bdos Interface, Bdos, Version 3.0 Nov, 1982'
                ;****************************************************************
                ;****************************************************************
                ;**                                                            **
                ;**  B a s i c   D i s k   O p e r a t i n g   S y s t e m    **
                ;**                                                           **
                ;**             C o n s o l e   P o r t i o n                  **
                ;**                                                            **
                ;****************************************************************
                ;****************************************************************
                ;
                ;       November 1982
                ;
                ;
                ;     Console handlers
                ;
                conin:
                      ;read console character to A
0353 2117077E36       lxi h,kbchar! mov a,m! mvi m,0! ora a! rnz
                      ;no previous keyboard character ready
035B C309FF           jmp coninf ;get character externally
                      ;ret
                ;
                conech:
035E 21CB06E5         LXI H,STA$RET! PUSH H
                CONECHO:
                      ;read character with echo
0362 CD5303CD8C       call conin! call echoc! JC CONECH1 ;echo character?
                       ;character must be echoed before return
036B F54FCDA004       push psw! mov c,a! call tabout! pop psw
0371 C9               RET
                CONECH1:
0372 CDA803C0         CALL TEST$CTLS$MODE! RNZ
0376 FE13C28103       CPI CTLS! JNZ CONECH2
037B CDE003C362       CALL CONBRK2! JMP CONECHO
                CONECH2:
0381 FE11CA6203       CPI CTLQ! JZ CONECHO
0386 FE10CA6203       CPI CTLP! JZ CONECHO
038B C9               RET
                ;
                echoc:
                      ;echo character if graphic
                      ;cr, lf, tab, or backspace
038C FE0DC8           cpi cr! rz ;carriage return?
038F FE0AC8           cpi lf! rz ;line feed?
0392 FE09C8           cpi tab! rz ;tab?
0395 FE08C8           cpi ctlh! rz ;backspace?
0398 FE20C9           cpi ' '! ret ;carry set if not graphic
                ;
                CONSTX:
039B 3A1707B7C2       LDA KBCHAR! ORA A! JNZ CONB1
03A2 CD06FFE601       CALL CONSTF! ANI 1! RET
                ;
```

```
                        if BANKED

                        SET$CTLS$MODE:
                                ;SET CTLS STATUS OR INPUT FLAG FOR QUEUE MANAGER
                                LXI H,QFLAG! MVI M,40H! XTHL! PCHL

                        endif
                        ;
                        TEST$CTLS$MODE:
                                ;RETURN WITH Z FLAG RESET IF CTL-S CTL-Q CHECKING DISABLED
03A8 473ACF1EE6                 MOV B,A! LDA CONMODE! ANI 2! MOV A,B! RET
                        ;
                        conbrk:         ;check for character ready
03B0 CDA803C29B                 CALL TEST$CTLS$MODE! JNZ CONSTX
03B6 3A1707B7C2                 lda Kbchar! ora a! jnz CONBRK1 ;skip if active Kbchar
                                        ;no active Kbchar, check external break
                                        ;DOES BIOS HAVE TYPE AHEAD?
                        if BANKED
                                        LDA TYPE$AHEAD! INR A! JZ CONSTX ;YES
                        endif
                                        ;CONBRKX CALLED BY CONOUT

                                CONBRKX:
                                        ;HAS CTL-S INTERCEPT BEEN DISABLED?
03BD CDA803C0                           CALL TEST$CTLS$MODE! RNZ ;YES
                                        ;DOES KBCHAR CONTAIN CTL-S?
03C1 3A1707FE13                         LDA KBCHAR! CPI CTLS! JZ CONBRK1 ;YES
                        if BANKED
                                        CALL SET$CTLS$MODE
                        endif
                                        ;IS A CHARACTER READY FOR INPUT?
03C9 CD06FF                             call constf
                        if BANKED
                                        POP H! MVI M,0
                        endif
03CC E601C8                             ani 1! rz ;NO
                                        ;character ready, read it
                        if BANKED
                                        CALL SET$CTLS$MODE
                        endif
03CF CD09FF                             call coninf
                        if BANKED
                                        POP H! MVI M,0
                        endif
                                CONBRK1:
03D2 FE13C2FB03                         cpi ctls! jnz conb0 ;check stop screen function
                                        ;DOES KBCHAR CONTAIN A CTL-S?
03D7 211707BEC2                         LXI H,KBCHAR! CMP M! JNZ CONBRK2 ;NO
03DE 3600                               MVI M,0 ; KBCHAR = 0
                                        ;found ctls, read next character
                                CONBRK2:

                        if BANKED
                                        CALL SET$CTLS$MODE
                        endif
03E0 CD09FF                             call coninf ;to A
```

```
                if BANKED
                               POP H! MVI M,0
                endif
03E3 FE03C2F103                cpi ctlc! JNZ CONBRK3
03E8 3ACF1EE608                LDA CONMODE! ANI 08H! JZ REBOOTX
03F0 AF                        XRA A
                       CONBRK3:
03F1 D611C8                    SUI CTLQ! RZ ; RETURN WITH A = ZERO IF CTLQ
03F4 3CCD1C04C3                INR A! CALL CONB3! JMP CONBRK2
                       conb0:
03FB 211707                    LXI H,KBCHAR

03FE 47                        MOV B,A
                               ;IS CONMODE(1) TRUE?
03FF 3ACF1E1FD2                LDA CONMODE! RAR! JNC $+7 ;NO
                               ;DOES KBCHAR = CTLC?
0406 3E03BEC8                  MVI A,CTLC! CMP M! RZ ;YES - RETURN
040A 78                        MOV A,B

040B FE11CA1904                CPI CTLQ! JZ CONB2
0410 FE10CA1904                CPI CTLP! JZ CONB2
                               ;character in accum, save it
0415 77                        MOV M,A
                       conb1:
                               ;return with true set in accumulator
0416 3E01C9                    mvi a,1! ret
                       CONB2:
0419 AF77C9                    XRA A! MOV M,A! RET
                       CONB3:
041C CC2504                    CZ TOGGLE$LISTCP
041F 0E07C40CFF                MVI C,7! CNZ CONOUTF
0424 C9                        RET
                ;
                TOGGLE$LISTCP:
                       ; IS PRINTER ECHO DISABLED?
0425 3ACF1EE614        LDA CONMODE! ANI 14H! JNZ TOGGLE$L1 ;YES
042D 21D41E3E01        LXI H,LISTCP! MVI A,1! XRA M! ANI 1
0435 77C9              MOV M,A! RET
                TOGGLE$L1:
0437 AFC9              XRA A! RET
                ;
                QCONOUTF:
                       ;DOES FX = INPUT?
0439 3ADF1E3DCA        LDA FX! DCR A! JZ CONOUTF ;YES
                       ;IS ESCAPE SEQUENCE DECODING IN EFFECT?
0440 78E608C20C        MOV A,B! ANI 8! JNZ SCONOUTF ;YES
0446 C30CFF            JMP CONOUTF
                ;
                conout:
                       ;compute character position/write console char from C
                       ;compcol = true if computing column position
0449 3A1507B7C2        lda compcol! ora a! jnz compout
                               ;write the character, then compute the column
                               ;write console character from C
                               ;B ^= 0 -> ESCAPE SEQUENCE DECODING
0450 3ACF1EE614                LDA CONMODE! ANI 14H! MOV B,A
```

```
0456 C5                       push b
                              ;CALL CONBRKX FOR OUTPUT FUNCTIONS ONLY
0457 3ADF1E3DC4               LDA FX! DCR A! CNZ CONBRKX
045E C1C5                     pop b! push b ;recall/save character
0460 CD3904                   call QCONOUTF ;externally, to console
0463 C1                       pop b
                              ;SKIP ECHO WHEN CONMODE & 14H ^= 0
0464 78B7C27204               MOV A,B! ORA A! JNZ COMPOUT
0469 C5                       push b ;recall/save character
                              ;may be copying to the list device
046A 3AD41EB7C4               lda listcp! ora a! cnz listf ;to printer, if so
0471 C1                       pop b ;recall the character
                      compout:
0472 79                       mov a,c ;recall the character
                              ;and compute column position
0473 21B71E                   lxi h,column ;A = char, HL = .column
0476 FE7FC8                   cpi rubout! rz ;no column change if nulls
0479 34                       inr m ;column = column + 1
047A FE20D0                   cpi ' '! rnc ;return if graphic
                              ;not graphic, reset column position
047D 35                       dcr m ;column = column - 1
047E 7EB7C8                   mov a,m! ora a! rz ;return if at zero
                              ;not at zero, may be backspace or end line
0481 79                       mov a,c ;character back to A
0482 FE08C28904               cpi ctlh! jnz notbacksp
                                      ;backspace character
0487 35                               dcr m ;column = column - 1
0488 C9                               ret
                              notbacksp:
                                      ;not a backspace character, eol?
0489 FE0DC0                           cpi cr! rnz ;return if not
                                      ;end of line, column = 0
048C 3600                             mvi m,0 ;column = 0
048E C9                       ret
               ;
               ctlout:
                      ;send C character with possible preceding up-arrow
048F 79CD8C03         mov a,c! call echoc ;cy if not graphic (or special case)
0493 D2A004           jnc tabout ;skip if graphic, tab, cr, lf, or ctlh
                              ;send preceding up arrow
0496 F50E5ECD49               push psw! mvi c,ctl! call conout ;up arrow
049C F1F640                   pop psw! ori 40h ;becomes graphic letter
049F 4F                       mov c,a ;ready to print
               if BANKED
                              call chk$column! rz
               endif
                              ;(drop through to tabout)
               ;
               tabout:
                      ;IS FX AN INPUT FUNCTION?
04A0 3ADF1E3DCA       LDA FX! DCR A! JZ TABOUT1 ;YES - ALWAYS EXPAND TABS FOR ECHO
                      ;HAS TAB EXPANSION BEEN DISABLED OR
                      ;ESCAPE SEQUENCE DECODING BEEN ENABLED?
04A7 3ACF1EE614       LDA CONMODE! ANI 14H! JNZ CONOUT ;YES
               TABOUT1:
                      ;expand tabs to console
```

```
04AF 79FE09C249         mov a,c! cpi tab! jnz conout ;direct to conout if not
                                ;tab encountered, move to next tab position
                        tab0:

                if BANKED
                                lda fx! cpi 1! jnz tab1
                                call chk$column! rz
                        tab1:
                endif

04B5 0E20CD4904                 mvi c,' '! call conout ;another blank
04BA 3AB71EE607                 lda column! ani 111b ;column mod 8 = 0 ?
04BF C2B504                     jnz tab0 ;back for another if not
04C2 C9                 ret
                ;
                ;
                backup:
                        ;back-up one screen position
04C3 CDCB04             call pctlh

                if BANKED
                        lda comchr! cpi ctla! rz
                endif

04C6 0E20CD0CFF         mvi c,' '! call conoutf
                ;       (drop through to pctlh)                         ;
                pctlh:
                        ;send ctlh to console without affecting column count
04CB 0E08C30CFF         mvi c,ctlh! jmp conoutf
                        ;ret
                ;
                crlfp:
                        ;print #, cr, lf for ctlx, ctlu, ctlr functions
                        ;then move to strtcol (starting column)
04D0 0E23CD4904         mvi c,'#'! call conout
04D5 CDE804             call crlf
                        ;column = 0, move to position strtcol
                        crlfp0:
04D8 3AB71E2116                 lda column! lxi h,strtcol
04DE BED0                       cmp m! rnc ;stop when column reaches strtcol
04E0 0E20CD4904                 mvi c,' '! call conout ;print blank
04E5 C3D804                     jmp crlfp0
                ;;
                ;
                crlf:
                        ;carriage return line feed sequence
04EB 0E0DCD4904         mvi c,cr! call conout! mvi c,lf! jmp conout
                        ;ret
                ;
                print:
                        ;print message until M(BC) = '$'
04F2 21D31E             LXI H,OUTDELIM
04F5 0ABEC8             ldax b! CMP M! rz ;stop on $
                                ;more to print
04F8 03C54F                     inx b! push b! mov c,a ;char to C
04FB CDA004                     call tabout ;another character printed
```

```
04FE C1C3F204                           pop b! jmp print
                        ;
                        QCONIN:

                        if BANKED
                                lhld apos! mov a,m! sta ctla$sw
                        endif
                                ;IS BUFFER ADDRESS = 0?
0502 2ABA1E7DB4                 LHLD CONBUFFADD! MOV A,L! ORA H! JZ CONIN ;YES
                                ;IS CHARACTER IN BUFFER < 5?

                        if BANKED
                                call qconinx ; mov a,m with bank 1 switched in
                        else
050A 7E                         MOV A,M
                        endif

050B 23                         INX H
050C B7C21305                   ORA A! JNZ QCONIN1 ; NO
0510 210000                     LXI H,0
                        QCONIN1:
0513 22BA1E22BC                 SHLD CONBUFFADD! SHLD CONBUFFLEN! RNZ ; NO
051A C35303                     JMP CONIN

                        if BANKED

                        chK$column:
                                lda conwidth! mov e,a! lda column! cmp e! ret
                        ;
                        expand:
                                xchg! lhld apos! xchg
                        expand1:
                                ldax d! ora a! rz
                                inx d! inx h! mov m,a! inr b! jmp expand1
                        ;
                        copy$xbuff:
                                mov a,b! ora a! rz
                                push b! mov c,b! push h! xchg! inx d
                                lxi h,xbuff
                                call move
                                mvi m,0! shld xpos
                                pop h! pop b! ret
                        ;
                        copy$cbuff:
                                lda ccpflgs+1! ral! rnc
                                lxi h,xbuff! lxi d,cbuff! inr c! jnz copy$cbuff1
                                xchg! mov a,b! ora a! rz
                                sta cbuff$len
                                push d! lxi b,copy$cbuff2! push b
                                mov b,a
                        copy$cbuff1:
                                inr b! mov c,b! jmp move
                        copy$cbuff2:
                                pop h! dcx h! mvi m,0! ret
                        ;
                        save$col:
```

```
		lda column! sta save$column! ret
	;
	clear$right:
		lda column! lxi h,ctla$column! cmp m! rnc
		mvi c,20h! call conout! jmp clear$right
	;
	reverse:
		lda save$column! lxi h,column! cmp m! rnc
		mvi c,ctlh! call conout! jmp reverse
	;
	chK$buffer$size:
		push b! push h
		lhld apos! mvi e,0
	cbs1:
		mov a,m! ora a! jz cbs2
		inr e! inx h! jmp cbs1
	cbs2:
		mov a,b! add e! cmp c
		push a! mvi c,7! cnc conoutf
		pop a! pop h! pop b! rc
		pop d! pop d! jmp readnx
	;
	refresh:
		lda ctla$sw! ora a! rz
		lda comchr! cpi ctla! rz
		cpi ctlf! rz
		cpi ctlw! rz
	refresh0:
		push h! push b
		call save$col
		lhld apos
	refresh1:
		mov a,m! ora a! jz refresh2
		mov c,a! call chK$column! jc refresh05
		mov a,e! sta column! jmp refresh2
	refresh05:
		push h! call ctlout
		pop h! inx h! jmp refresh1
	refresh2:
		lda column! sta new$ctla$col
	refresh3:
		call clear$right
		call reverse
		lda new$ctla$col! sta ctla$column
		pop b! pop h! ret
	;
	init$apos:
		lxi h,aposi! shld apos
		xra a! sta ctla$sw
		ret
	;
	init$xpos:
		lxi h,xbuff! shld xpos! ret
	;
	set$ctla$column:
		lxi h,ctla$sw! mov a,m! ora a! rnz
```

```
                                inr m! lda column! sta ctla$column! ret
                      ;
                      readi:
                                call chk$column! cnc crlf
                                lda cbuff$len! mov b,a
                                mvi c,0! call copy$cbuff
                      else

                      readi:
051D 7AB3C22D05                 MOV A,D! ORA E! JNZ READ
0522 2AD81E22DB                 LHLD DMAAD! SHLD INFO
0528 232322BA1E                 INX H! INX H! SHLD CONBUFFADD
                      endif

                      read:     ;read to info address (max length, current length, buffer)

                      if BANKED
                                call init$xpos
                                call init$apos
                      readx:
                                call refresh
                                xra a! sta ctlw$sw
                      readx1:

                      endif

052D 3E0132DF1E                 MVI A,1! STA FX
0532 3AB71E3216                 lda column! sta strtcol ;save start for ctl-x, ctl-h
0538 2ADB1E4E23                 lhld info! mov c,m! inx h! push h
053E AF47321807                 XRA A! MOV B,A! STA SAVEPOS
0543 B9C24805                   CMP C! JNZ $+4
0547 0C                         INR C
                                ;B = current buffer length,
                                ;C = maximum buffer length,
                                ;HL= next to fill - 1
                                readnx:
                                          ;read next character, BC, HL active
0548 C5E5                                 push b! push h ;blen, cmax, HL saved
                                          readn0:

                      if BANKED
                                                    lda ctlw$sw! ora a! cz qconin
                      nxtline:
                                                    sta comchr
                      else
054A CD0205                                         CALL QCONIN ;next char in A
                      endif

                                                    ;ani 7fh ;mask parity bit
054D E1C1                                           pop h! pop b ;reactivate counters
054F FE0DCA5706                                     cpi cr! jz readen ;end of line?
0554 FE0ACA5706                                     cpi lf! jz readen ;also end of line

                      if BANKED
                                                    cpi ctlf! jnz not$ctlf
                                          do$ctlf:
```

CP/M - PLUS V 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
		call chk$column! dcr e! cmp e! jnc readnx
	do$ctlf0:
		xchg! lhld apos! mov a,m! ora a! jz ctlw$115
		inx h! shld apos! xchg! jmp notr
	not$ctlf:
		cpi ctlw! jnz not$ctlw
	do$ctlw:
		xchg! lhld apos! mov a,m! ora a! jz ctlw$11
		xchg! call chk$column! dcr e! cmp e! xchg! jc ctlw$10
		xchg! call refresh0! xchg! jmp ctlw$113
	ctlw$10:
		lhld apos! mov a,m
		inx h! shld apos! jmp ctlw$13
	ctlw$11:
		lxi h,ctla$sw! mov a,m! mvi m,0
		ora a! jz ctlw$12
	ctlw$113:
		lxi h,ctlw$sw! mvi m,0
	ctlw$115:
		xchg! jmp readnx
	ctlw$12:
		lda ctlw$sw! ora a! jnz ctlw$125
		mov a,b! ora a! jnz ctlw$115
		call init$xpos
	ctlw$125:
		lhld xpos! mov a,m! ora a
		sta ctlw$sw! jz ctlw$115
		inx h! shld xpos
	ctlw$13:
		lxi h,ctlw$sw! mvi m,ctlw
		xchg! jmp notr
	not$ctlw:
		cpi ctla! jnz not$ctla
	do$ctla:
		;do we have any characters to back over?
		lda strtcol! mov d,a! lda column! cmp d
		jz readnx
		sta compcol ;COL > 0
		mov a,b! ora a! jz linelen
		;characters remain in buffer, backup one
		dcr b ;remove one character
		;compcol > 0 marks repeat as length compute
		;backup one position in xbuff
		push h
		call set$ctla$column
		pop d
		lhld apos! dcx h
		shld apos! ldax d! mov m,a! xchg! jmp linelen
	not$ctla:
		cpi ctlb! jnz not$ctlb
	do$ctlb:
		lda save$pos! cmp b! jnz ctlb$10
		mvi a,ctlw! sta ctla$sw
		sta comchr! jmp do$ctlw
	ctlb$10:
		xchg! lhld apos! inr b
```

```
                                        ctlb$l1:
                                                dcr b! lda save$pos! cmp b! jz ctlb$l2
                                                dcx h! ldax d! mov m,a! dcx d! jmp ctlb$l1
                                        ctlb$l2:
                                                shld apos
                                                push b! push d
                                                call set$ctla$column
                                        ctlb$l3:
                                                lda column! mov b,a
                                                lda strtcol! cmp b! jz read$n0
                                                mvi c,ctlh! call conout! jmp ctlb$l3
                                        not$ctlb:
                                                cpi ctlK! jnz not$ctlk
                                                xchg! lxi h,aposi! shld apos
                                                xchg! call refresh
                                                jmp readnx
                                        not$ctlk:
                                                cpi ctlg! jnz not$ctlg
                                                lda ctla$sw! ora a! jz readnx
                                                jmp do$ctlf0
                                        not$ctlg:
                    endif

0559 FE08C27C05                                 cpi ctlh! jnz noth ;backspace?
055E 3ACA1E3CCA                                 LDA CTLH$ACT! INR A! JZ DO$RUBOUT
                                        DO$CTLH:
                                                ;do we have any characters to back over?
0565 3A1607573A                                 LDA STRTCOL! MOV D,A! LDA COLUMN! CMP D
056D CA4805                                     jz readnx
0570 321507                                     STA COMPCOL ;COL > 0
0573 78B7CA7905                                 MOV A,B! ORA A! JZ $+4
                                                ;characters remain in buffer, backup one
0578 05                                         dcr b ;remove one character
                                                ;compcol > 0 marks repeat as length compute
0579 C3E405                                     jmp linelen ;uses same code as repeat
                                        noth:
                                                ;not a backspace
057C FE7FC29305                                 cpi rubout! jnz notrub ;rubout char?
0581 3ACB1E3CCA                                 LDA RUBOUT$ACT! INR A! JZ DO$CTLH
                                        DO$RUBOUT:
                    if BANKED
                                                mvi a,rubout! sta comchr
                                                lda ctla$sw! ora a! jnz do$ctlh
                    endif
                                                ;rubout encountered, rubout if possible
0588 78B7CA4805                                 mov a,b! ora a! jz readnx ;skip if len=0
                                                ;buffer has characters, resend last char
058D 7E052B                                     mov a,m! dcr b! dcx h ;A = last char
                                                ;blen=blen-1, next to fill - 1 decremented
0590 C33806                                     jmp rdech1 ;act like this is an echo
                                        notrub:
                                                ;not a rubout character, check end line
0593 FE05C2A805                                 cpi ctle! jnz note ;physical end line?
                                                ;yes, save active counters and force eol
0598 C578321807                                 push b! MOV A,B! STA SAVE$POS
059D E5                                         push h
```

```
               if BANKED
                                      lda ctla$sw! ora a! cnz clear$right
               endif
059E CDE804                           call crlf
               if BANKED
                                      call refresh
               endif
05A1 AF321607                         xra a! sta strtcol ;start position = 00
05A5 C34A05                           jmp readn0 ;for another character
                              note:
                                      ;not end of line, list toggle?
05A8 FE10C2B805                       cpi ctlp! jnz notp ;skip if not ctlp
                                      ;list toggle - change parity
05AD E5                               push h ;save next to fill - 1
05AE C5                               PUSH B
05AF AFCD1C04                         XRA A! CALL CONB3
05B3 C1                               POP B
05B4 E1C34805                         pop h! jmp readnx ;for another char
                              notp:
                                      ;not a ctlp, line delete?
05B8 FE18C2CF05                       cpi ctlx! jnz notx
05BD E1                               pop h ;discard start position
                                      ;loop while column > strtcol
                                      backx:
05BE 3A160721B7                               lda strtcol! lxi h,column
               if BANKED
                                              cmp m! jc backx1
                                              lhld apos! mov a,m! ora a! jnz readx
                                              jmp read
                                          backx1:
               else
05C4 BED22D05                                 cmp m! jnc read ;start again
               endif
05C8 35                                       dcr m ;column = column - 1
05C9 CDC304                                   call backup ;one position
05CC C3BE05                                   jmp backx
                              notx:
                                      ;not a control x, control u?
                                      ;not control-X, control-U?
05CF FE15C2DB05                       cpi ctlu! jnz notu ;skip if not
               if BANKED
                                      xthl! call copy$xbuff! xthl
               endif
                                      ;delete line (ctlu)
                              do$ctlu:
05D4 CDD004                           call crlfp ;physical eol
05D7 E1                               pop h ;discard starting position
05D8 C32D05                           jmp read ;to start all over
                              notu:
                                      ;not line delete, repeat line?
05DB FE12C22306                       cpi ctlr! jnz notr
05E0 AF321807                         XRA A! STA SAVEPOS
               if BANKED
                                      xchg! call init$apos! xchg
                                      mov a,b! ora a! jz do$ctlu
                                      xchg! lhld apos! inr b
                              ctlr$11:
                                      dcr b! jz ctlr$12
                                      dcx h! ldax d! mov m,a! dcx d
                                      jmp ctlr$11
                              ctlr$12:
                                      shld apos! push b! push d
                                      call crlfp! mvi a,ctlw! sta ctlw$sw
                                      sta ctla$sw! jmp readn0
               endif
                              linelen:
                                      ;repeat line, or compute line len (ctlh)
                                      ;if compcol > 0
05E4 C5CDD004                         push b! call crlfp ;save line length
05E9 C1E1E5C5                         pop b! pop h! push h! push b
                                      ;bcur, cmax active, beginning buff at HL
                              rep0:
05EC 78B7CA0706                       mov a,b! ora a! jz rep1 ;count len to 00
05F1 234E                             inx h! mov c,m ;next to print
05F3 05                               DCR B
05F4 D1D57A9057                       POP D! PUSH D! MOV A,D! SUB B! MOV D,A
05F9 C5E5                             push b! push h ;count length down
05FB 3A1807BADC                       LDA SAVEPOS! CMP D! CC CTLOUT
0602 E1C1                             pop h! pop b ;recall remaining count
0604 C3EC05                           jmp rep0 ;for the next character
                              rep1:
                                      ;end of repeat, recall lengths
                                      ;original BC still remains pushed
0607 E5                               push h ;save next to fill
0608 3A1507B7                         lda compcol! ora a ;>0 if computing length
060C CA4A05                           jz readn0 ;for another char if so
                                      ;column position computed for ctlh
060F 21B71E96                         lxi h,column! sub m ;diff > 0
0613 321507                           sta compcol ;count down below
                                      ;move back compcol-column spaces
                              backsp:
                                      ;move back one more space
0616 CDC304                           call backup ;one space
0619 21150735                         lxi h,compcol! dcr m
061D C21606                           jnz backsp
               if BANKED
                                      call refresh
               endif
0620 C34A05                           jmp readn0 ;for next character
                              notr:
                                      ;not a ctlr, place into buffer
                                      ;IS BUFFER FULL?
0623 F5                               PUSH A
0624 78B9DA3406                       MOV A,B! CMP C! JC RDECHO ;NO
                                      ;DISCARD CHARACTER AND RING BELL
0629 F1C5E5                           POP A! PUSH B! PUSH H
062C 0E07CD0CFF                       MVI C,7! CALL CONOUTF! JMP READNO
                              RDECHO:

               if BANKED
                                      lda comchr! cpi ctlg! jz rdech05
                                      lda ctla$sw! ora a! cnz chk$buffer$size
```

```
                                rdech05:
                endif

0634 F1                                 POP A
0635 2377                               inx h! mov m,a ;character filled to mem
0637 04                                 inr b ;blen = blen + 1
                                rdech1:
                                        ;look for a random control character
0638 C5E5                               push b! push h ;active values saved
063A 4F                                 mov c,a ;ready to print
                if BANKED
                                        call save$col
                endif
063B CD8F04                             call ctlout ;may be up-arrow C
063E E1C1                               pop h! pop b
                if BANKED
                                        lda comchr! cpi ctlg! jz do$ctlh
                                        cpi rubout! jz rdech2
                                        call refresh
                                rdech2:
                endif
0640 3ACF1EE608                         LDA CONMODE! ANI 08H! JNZ NOTC
0648 7E                                 mov a,m ;recall char
0649 FE03                               cpi ctlc ;set flags for reboot test
064B 78                                 mov a,b ;move length to A
064C C25406                             jnz notc ;skip if not a control c
064F FE01                               cpi 1 ;control C, must be length 1
0651 CAC202                             jz REBOOTX ;reboot if blen = 1
                                        ;length not one, so skip reboot
                                notc:
                                        ;not reboot, are we at end of buffer?
                if BANKED
                                        cmp c! jnc buffer$full
                else
0654 C34805                             jmp readnx ;go for another if not
                endif

                if BANKED
                                        push b! push h
                                        call chk$column! jc readn0
                                        lda ctla$sw! ora a! jz do$new$line
                                        lda comchr! cpi ctlw! jz back$one
                                        cpi ctlf! jz back$one

                                do$newline:
                                        mvi a,ctle! jmp nxtline

                                back$one:
                                        ;back up to previous character
                                        pop h! pop b
                                        dcr b! xchg
                                        lhld apos! dcx h! shld apos
                                        ldax d! mov m,a! xchg! dcx h
                                        push b! push h! call reverse
                                        ;disable ctlb or ctlw
                                        xra a! sta ctlw$sw! jmp readn0
```

```

                              buffer$full:
                                      xra a! sta ctlw$sw! jmp readnx
               endif
                              readen:
                                      ;end of read operation, store blen
               if BANKED
                                      call expand
               endif
0657 E170                             pop h! mov m,b ;M(current len) = B
               if BANKED
                                      push b
                                      call copy$xbuff
                                      pop b
                                      mvi c,0ffh! call copy$cbuff
               endif
0659 21000022BA                       LXI H,0! SHLD CONBUFFADD
065F 0E0DC34904                       mvi c,cr! jmp conout ;return carriage
                                      ;ret
               ;
035E =         func1  equ     CONECH
                      ;return console character with echo
               ;
04A0 =         func2: equ     tabout
                      ;write console character with tab expansion
               ;
               func3:
                      ;return reader character
0664 CD15FF           call readerf
0667 C3CB06           jmp sta$ret
               ;
               ;func4:        equated to punchf
                      ;write punch character
               ;
               ;func5:        equated to listf
                      ;write list character
                      ;write to list device
               ;
               func6:
                      ;direct console i/o - read if 0ffh
066A 793CCA8306       mov a,c! inr a! jz dirinp ;0ffh => 00h, means input mode
066F 3CCA7A06                 inr a! JZ DIRSTAT ;0feh => direct STATUS function
0673 3CCA8806                 INR A! JZ DIRINP1 ;0fdh => direct input, no status
0677 C30CFF                   JMP CONOUTF
                      DIRSTAT:
                              ;0feH in C for status
067A CD9B03C226               CALL CONSTX! JNZ LRET$EQ$FF! JMP STA$RET
                      dirinp:
0683 CD9B03                   CALL CONSTX ;status check
0686 B7C8                     ora a! RZ ;skip, return 00 if not ready
                              ;character is ready, get it
                      dirinp1:
0688 CD5303                   call CONIN ;to A
068B C3CB06                   jmp sta$ret
               ;
               func7:
```

```
068E CD36FF             call auxinstf
0691 C3CB06             jmp sta$ret
                ;
                func8:
0694 CD39FF             call auxoutstf
0697 C3CB06             jmp sta$ret
                ;
                func9:
                        ;write line until $ encountered
069A EB                 xchg    ;was lhld info
069B 4D44               mov c,l! mov b,h ;BC=string address
069D C3F204             jmp print ;out to console

051D =          func10 equ     readi
                        ;read a buffered console line

                func11:
                        ;IS CONMODE(1) TRUE?
06A0 3ACF1E1FD2         LDA CONMODE! RAR! JNC NORMAL$STATUS ;NO
                        ;CTL-C ONLY STATUS CHECK
                if BANKED
                        LXI H,QFLAG! MVI M,80H! PUSH H
                endif
06A7 21C506E5           LXI H,CTLC$STAT$RET! PUSH H
                        ;DOES KBCHAR = CTL-C?
06AB 3A1707FE03         LDA KBCHAR! CPI CTLC! JZ CONB1 ;YES
                        ;IS THERE A READY CHARACTER?
06B3 CD06FFB7C8         CALL CONSTF! ORA A! RZ ;NO
                        ;IS THE READY CHARACTER A CTL-C?
06B8 CD09FFFE03         CALL CONINF! CPI CTLC! JZ CONB0 ;YES
06C0 321707AFC9         STA KBCHAR! XRA A! RET

                CTLC$STAT$RET:

                if BANKED
                        CALL STA$RET
                        POP H! MVI M,0! RET
                else
06C5 C3CB06             JMP STA$RET
                endif

                NORMAL$STATUS:
                        ;check console status
06C8 CDB003             call conbrk
                        ;(drop through to sta$ret)
                sta$ret:
                        ;store the A register to aret
06CB 327A07             sta aret
                func$ret:                                                  ;
06CE C9                 ret ;jmp goback (pop stack for non cp/m functions)
                ;
                setlret1:
                        ;set lret = 1
06CF 3E01C3CB06         mvi a,1! jmp sta$ret                               ;
                ;
                FUNC109:                        ;GET/SET CONSOLE MODE
```

```
                      ;DOES DE = 0FFFFH?
06D4 7AA33C           MOV A,D! ANA E! INR A
06D7 2ACF1ECAC3       LHLD CONMODE! JZ STHL$RET ;YES - RETURN CONSOLE MODE
06DD EB22CF1EC9       XCHG! SHLD CONMODE! RET ;NO - SET CONSOLE MODE
               ;
               FUNC110:                       ;GET/SET FUNCTION 9 DELIMITER
06E2 21D31E           LXI H,OUT$DELIM
                      ;DOES DE = 0FFFFH?
06E5 7AA33C           MOV A,D! ANA E! INR A
06E8 7ECACB06         MOV A,M! JZ STA$RET ;YES - RETURN DELIMITER
06EC 73C9             MOV M,E! RET ;NO - SET DELIMITER
               ;
               FUNC111:                       ;PRINT BLOCK TO CONSOLE
               FUNC112:                       ;LIST BLOCK
06EE EB5E235623       XCHG! MOV E,M! INX H! MOV D,M! INX H
06F3 4E2346EB         MOV C,M! INX H! MOV B,M! XCHG
                      ;HL = ADDR OF STRING
                      ;BC = LENGTH OF STRING
               BLK$OUT:
06F7 78B1C8           MOV A,B! ORA C! RZ
06FA C5E54E           PUSH B! PUSH H! MOV C,M
06FD 3ADF1EFE6F       LDA FX! CPI 111! JZ BLK$OUT1
0705 CD0FFFC30E       CALL LISTF! JMP BLK$OUT2
               BLK$OUT1:
070B CDA004           CALL TABOUT
               BLK$OUT2:
070E E123C10B         POP H! INX H! POP B! DCX B
0712 C3F706           JMP BLK$OUT

FF0C =         SCONOUTF       EQU     CONOUTF

               ;
               ;      data areas
               ;
0715 00        compcol:db     0       ;true if computing column position
0716 00        strtcol:db     0       ;starting column position after read

               if not BANKED

0717 00        Kbchar:        db      0       ;initial Key char = 00

               endif

0718 00        SAVEPOS:DB     0       ;POSITION IN BUFFER CORRESPONDING TO
                                      ;BEGINNING OF LINE
               if BANKED

               comchr:                db      0
               cbuff$len:     db      0
               cbuff:         ds      256
                              db      0
               xbuff:         db      0
                              ds      354
               aposi:         db      0
               xpos:          dw      0
               apos:          dw      0
```

```
                ctla$sw:        db      0
                ctlw$sw:        db      0
                save$column:    db      0
                ctla$column:    db      0
                new$ctla$col:   db      0

                endif

                ;       end of BDOS Console module
                ;
                ;**********************************************************************
                ;*****************************************************************
                ;
                ;       Error Messages

                if BANKED

                md      equ     0

                else

0024 =          md      equ     24h

                endif

0719 43502F4D20dskmsg:  db      'CP/M Error On '
0727 203A2024   dskerr: db      ' : ',md
072B 4469736B20permsg:  db      'Disk I/O',md
0734 496E76616Cselmsg:  db      'Invalid Drive',md
0742 526561642Frofmsg:  db      'Read/Only File',md
0751 526561642Frodmsg:  db      'Read/Only Disk',md

                if not MPM

                passmsg:

                if BANKED
                        db      'Password Error',md
                endif

                fxstsmsg:
0760 46696C6520         db      'File Exists',md

                wildmsg:
076C 3F20696E20         db      '? in Filename',md

                endif
                if MPM

                setlret1:
                        mvi a,1
                sta$ret:
                        sta     aret
                func$ret:
                        ret
                entsp:  ds      2

                endif

                ;*****************************************************************
                ;*****************************************************************
                ;
                ;       common values shared between bdosi and bdos

                if MPM

                usrcode:db      0       ; current user number

                endif

077A            aret:  ds       2       ; address value to return
077A =          lret   equ      aret    ; low(aret)

                ;*****************************************************************
                ;*****************************************************************
                ;**                                                             **
                ;**  b a s i c   d i s k   o p e r a t i n g   s y s t e m     **
                ;**                                                             **
                ;*****************************************************************
                ;*****************************************************************

                ;       literal constants

00FF =          true   equ      0ffh    ; constant true
0000 =          false  equ      000h    ; constant false
FFFF =          enddir equ      0ffffh  ; end of directory
0001 =          byte   equ      1       ; number of bytes for 'byte' type
0002 =          word   equ      2       ; number of bytes for 'word' type

                ;       fixed addresses in low memory

005C =          tfcb   equ      005ch   ; default fcb location
0080 =          tbuff  equ      0080h   ; default buffer location

                ;       error message handlers

                rod$error:
                        ; report read/only disk error
077C 0E02C38D07         mvi c,2! jmp goerr

                rof$error:
                        ; report read/only file error
0781 0E03C38D07         mvi c,3! jmp goerr

                sel$error:
                        ; report select error
0786 0E04               mvi c,4
                        ; Invalidate curdsk to force select call
                        ; at next curselect call
0788 3EFF32CB1C         mvi a,0ffh! sta curdsk

                goerr:
                        ; hl = .errorhandler, call subroutine
078D 612EFF227A         mov h,c! mvi l,0ffh! shld aret

                if MPM
                        call test$error$mode! jnz rtn$phy$errs
                        mov a,c! lxi h,pererr-2! jmp bdos$jmp
                else

                goerr1:
0793 3ACC1C32ED         lda adrive! sta errdrv
0799 3AE71E3CC4         lda error$mode! inr a! cnz error
                endif

                rtn$phy$errs:

                if MPM
                        lda lock$shell! ora a! jnz lock$perr
                endif

                        ; Return 0ffffh if fx = 27 or 31

07A0 3ADF1E             lda fx
07A3 FE1BCA5E1C         cpi 27! jz goback0
07A8 FE1FCA5E1C         cpi 31! jz goback0
07AD C3641C             jmp goback

                if MPM

                test$error$mode:
                        lxi d,pname+4
                test$error$mode1:
                        call rlr! dad d
                        mov a,m! ani 80h! ret
                endif

                if BANKED

                set$copy$cr$only:
                        lda copy$cr$init! sta copy$cr$only! ret

                reset$copy$cr$only:
                        xra a! sta copy$cr$init! sta copy$cr$only! ret

                endif

                bde$e$bde$m$hl:
07B0 7B955F             mov a,e! sub l! mov e,a
07B3 7A9C57             mov a,d! sbb h! mov d,a
07B6 D005C9             rnc! dcr b! ret

                bde$e$bde$p$hl:
07B9 7B855F             mov a,e! add l! mov e,a
07BC 7A8C57             mov a,d! adc h! mov d,a
07BF D004C9             rnc! inr b! ret

                shl3bv:
```

```
07C2 0C                 inr c
                shl3bv1:
07C3 0DC8               dcr c! rz
07C5 298FC3C307         dad h! adc a! jmp shl3bv1

                incr$rr:
07CA CDF509             call get$rra
07CD 34C0               inr m! rnz
07CF 2334C0             inx h! inr m! rnz
07D2 2334C9             inx h! inr m! ret

                save$rr:
07D5 CDDE07EB           call save$rr2! xchg
                save$rr1:
07D9 0E03C3F507         mvi c,3! jmp move ; ret
                save$rr2:
07DE CDF50911D0         call get$rra! lxi d,save$ranr! ret

                reset$rr:
07E5 CDDE07C3D9         call save$rr2! jmp save$rr1 ; ret

                compare:
07EB 1ABEC0             ldax d! cmp m! rnz
07EE 23130DC8           inx h! inx d! dcr c! rz
07F2 C3EB07             jmp compare

                ;
                ;       local subroutines for bios interface
                ;

                move:
                        ; Move data length of length c from source de to
                        ; destination given by hl
07F5 0C                 inr c ; in case it is zero
                        move0:
07F6 0DC8                       dcr c! rz ; more to move
07F8 1A77                       ldax d! mov m,a ; one byte moved
07FA 1323                       inx d! inx h ; to next byte
07FC C3F607                     jmp move0

                selectdisk:
                        ; Select the disk drive given by register D, and fill
                        ; the base addresses curtrka - alloca, then fill
                        ; the values of the disk parameter block
07FF 4A                 mov c,d ; current disk# to c
                        ; lsb of e = 0 if not yet logged - in
0800 CD1BFF             call seldskf ; hl filled by call
                        ; hl = 0000 if error, otherwise disk headers
0803 7CB5C8             mov a,h! ora l! rz ; Return with C flag reset if select error
                                ; Disk header block address in hl
0806 5E235623                   mov e,m! inx h! mov d,m! inx h ; de=.tran
080A 228E1C2323                 shld cdrmaxa! inx h! inx h ; .cdrmax
080F 22901C2323                 shld curtrka! inx h! inx h ; hl=.currec
0814 22921C2323                 shld curreca! inx h! inx h ; hl=.buffa
0819 2322941C23                 inx h! shld drvlbla! inx h
081E 22961C2323                 shld lsn$add! inx h! inx h
```

```
                              ; de still contains .tran
0823 EB22BF1C                 xchg! shld tranv ; .tran vector
0827 21981C                   lxi h,dpbaddr ; de= source for move, hl=dest
082A 0E0DCDF507               mvi c,addlist! call move ; addlist filled
                              ; Now fill the disk parameter block
082F 2A981CEB                 lhld dpbaddr! xchg ; de is source
0833 21A51C                   lxi h,sectpt ; hl is destination
0836 0E11CDF507               mvi c,dpblist! call move ; data filled
                              ; Now set single/double map mode
083B 2AAA1C                   lhld maxall ; largest allocation number
083E 7C                       mov a,h ; 00 indicates < 255
083F 21C51C36FF               lxi h,single! mvi m,true ; Assume a=00
0844 B7CA4A08                 ora a! jz retselect
                              ; high order of maxall not zero, use double dm
0848 3600                     mvi m,false
                      retselect:
                              ; C flag set indicates successful select
084A 37C9                     stc! ret

               home:
                      ; Move to home position, then offset to start of dir
084C CD18FF           call homef
084F AF               xra a ; constant zero to accumulator
0850 2A901C7723       lhld curtrka! mov m,a! inx h! mov m,a ; curtrk=0000
0856 2A921C7723       lhld curreca! mov m,a! inx h! mov m,a ; currec=0000
085C 2377             inx h! mov m,a ; currec high byte=00

               if MPM
                      lxi h,0! shld dblk ; dblk = 0000
               endif

085E C9               ret

               rdbuff:
                      ; Read buffer and check condition
085F 3E0132D61C       mvi a,1! sta readf$sw
0864 CD27FF           call readf ; current drive, track, sector, dma
0867 C37108           jmp diocomp ; Check for i/o errors

               wrbuff:
                      ; Write buffer and check condition
                      ; write type (wrtype) is in register c
086A AF32D61C         xra a! sta readf$sw
086E CD2AFF           call writef ; current drive, track, sector, dma
               diocomp: ; Check for disk errors
0871 B7C8             ora a! rz
0873 4F               mov c,a
0874 CD8208           call chk$media$flag
0877 79               mov a,c
0878 FE03DA8D07       cpi 3! jc goerr
087D 0E01C38D07       mvi c,1! jmp goerr

               chk$media$flag:
                      ; A = 0ffh -> media changed
0882 3CC0             inr a! rnz

```

```
                if BANKED
                       ; Handle media changes as I/O errors for
                       ; permanent drives
                       call chksiz$eq$8000h! rz
                endif

                       ; BIOS says media change occurred
                       ; Is disk logged-in?
0884 2A881CCDDB        lhld dlog! call test$vector! mvi c,1! rz ; no - return error
088D CD700B            call media$change
0890 E1                pop h ; Discard return address
                       ; Was this a flush operation (fx = 48)?
0891 3ADF1EFE30        lda fx! cpi 48! rz ; yes
                       ; Is this a flush to another drive?
0897 21CC1C3AC6        lxi h,adrive! lda seldsk! cmp m! jnz reset$relog
                       ; Bail out if fx = read, write, close, or search next
08A1 CDD10B            call chk$exit$fxs
                       ; Is this a directory read operation?
08A4 3AD61CB7C0        lda readf$sw! ora a! rnz ; yes
                       ; Error - directory write operation
08A9 0E02C38D07        mvi c,2! jmp goerr ; Return disk read/only error

                reset$relog:
                       ; Reset relog if flushing to another drive
08AE AF32DE1EC9        xra a! sta relog! ret

                if BANKED

                chksiz$eq$8000h:
                       ; Return with Z flag set if drive permanent
                       ; with no checksum vector
                       lhld chksiz! mvi a,80h! cmp h! rnz
                       xra a! cmp l! ret

                endif

                seekdir:
                       ; Seek the record containing the current dir entry

                if MPM
                       lxi d,0ffffh ; mask = ffff
                       lhld dblk! mov a,h! ora l! jz seekdir1
                       lda blkmsk! mov e,a! xra a! mov d,a ; mask = blkmsk
                       lda blkshf! mov c,a! xra a
                       call shl3bv ; ahl = shl(dblk,blkshf)
                seekdir1:
                       push h! push a ; Save ahl
                endif

08B3 2AE11E            lhld dcnt ; directory counter to hl
08B6 0E02CD690A        mvi c,dskshf! call hlrotr ; value to hl
08BB 22B61C            shld drec

                if MPM

                ;      arecord = shl(dblk,blkshf) + shr(dcnt,dskshf) & mask
```

```

                                mov a,l! ana e! mov l,a ; dcnt = dcnt & mask
                                mov a,h! ana d! mov h,a
                                pop b! pop d! call bde$e$bde$p$hl

                        else
     08BE 0600EB                mvi b,0! xchg
                        endif

                        set$arecord:
     08C1 21CD1C                lxi h,arecord
     08C4 7323722370            mov m,e! inx h! mov m,d! inx h! mov m,b
     08C9 C9                    ret

                        seek:
                                ; Seek the track given by arecord (actual record)

     08CA 2A901C4E23            lhld curtrka! mov c,m! inx h! mov b,m ; bc = curtrk
     08D0 C5                    push b ; s0 = curtrk
     08D1 2A921C5E23            lhld curreca! mov e,m! inx h! mov d,m
     08D7 2346                  inx h! mov b,m ; bde = currec
     08D9 2ACD1C3ACF            lhld arecord! lda arecord+2! mov c,a ; chl = arecord
                        seek0:
     08E0 7D937C9A79            mov a,l! sub e! mov a,h! sbb d! mov a,c! sbb b
     08E6 E5                    push h ; Save low(arecord)
     08E7 D2F708                jnc seek1 ; if arecord >= currec then go to seek1
     08EA 2AA51CCDB0            lhld sectpt! call bde$e$bde$m$hl ; currec = currec - sectpt
     08F0 E1E32BE3              pop h! xthl! dcx h! xthl ; curtrk = curtrk - 1
     08F4 C3E008                jmp seek0
                        seek1:
     08F7 2AA51CCDB9            lhld sectpt! call bde$e$bde$p$hl ; currec = currec + sectpt
     08FD E1                    pop h ; Restore low(arecord)
     08FE 7D937C9A79            mov a,l! sub e! mov a,h! sbb d! mov a,c! sbb b
     0904 DA0E09                jc seek2 ; if arecord < currec then go to seek2
     0907 E323E3                xthl! inx h! xthl ; curtrk = curtrk + 1
     090A E5                    push h ; save low (arecord)
     090B C3F708                jmp seek1
                        seek2:
     090E E3E5                  xthl! push h ; hl,s0 = curtrk, s1 = low(arecord)
     0910 2AA51CCDB0            lhld sectpt! call bde$e$bde$m$hl ; currec = currec - sectpt
     0916 E1D5C5E5              pop h! push d! push b! push h ; hl,s0 = curtrk,
                                ; s1 = high(arecord,currec), s2 = low(currec),
                                ; s3 = low(arecord)
     091A EB2AB21C19            xchg! lhld offset! dad d
     091F 444D22FE1C            mov b,h! mov c,l! shld track
     0924 CD1EFF                call settrkf ; call bios settrk routine
                                ; Store curtrk
     0927 D12A901C73            pop d! lhld curtrka! mov m,e! inx h! mov m,d
                                ; Store currec
     092E C1D1                  pop b! pop d!
     0930 2A921C7323            lhld curreca! mov m,e! inx h! mov m,d
     0936 2370                  inx h! mov m,b ; currec = bde
     0938 C1                    pop b ; bc = low(arecord), de = low(currec)
     0939 79936F                mov a,c! sub e! mov l,a ; hl = bc - de
     093C 789A67                mov a,b! sbb d! mov h,a
     093F CD5B09                call shr$physhf
```

```
0942 444D                  mov b,h! mov c,l

0944 2ABF1CEB              lhld tranv! xchg ; bc=sector#, de=.tran
0948 CD30FF                call sectran ; hl = tran(sector)
094B 4D44                  mov c,l! mov b,h ; bc = tran(sector)
094D 22001D                shld sector
0950 CD21FF                call setsecf ; sector selected
0953 2A8A1C4D44            lhld curdma! mov c,l! mov b,h! jmp setdmaf
                           ; ret
                   shr$physhf:
095B 3AB41C4FC3            lda physhf! mov c,a! jmp hlrotr

                   ;       file control block (fcb) constants

00E5 =             empty  equ     0e5h    ; empty directory entry
007F =             lstrec equ     127     ; last record# on extent
0080 =             recsiz equ     128     ; record size
0020 =             fcblen equ     32      ; file control block size
0004 =             dirrec equ     recsiz/fcblen   ; directory fcbs / record
0002 =             dskshf equ     2       ; log2(dirrec)
0003 =             dskmsk equ     dirrec-1
0005 =             fcbshf equ     5       ; log2(fcblen)

000C =             extnum equ     12      ; extent number field
001F =             maxext equ     31      ; largest extent number
000D =             ubytes equ     13      ; unfilled bytes field
000E =             modnum equ     14      ; data module number

0040 =             maxmod equ     64      ; largest module number

0080 =             fwfmsk equ     80h     ; file write flag is high order modnum
000F =             namlen equ     15      ; name length
000F =             reccnt equ     15      ; record count field
0010 =             dskmap equ     16      ; disk map field
001F =             lstfcb equ     fcblen-1
0020 =             nxtrec equ     fcblen
0021 =             ranrec equ     nxtrec+1; random record field (2 bytes)

                   ;       reserved file indicators

0009 =             rofile equ     9       ; high order of first type char
000A =             invis  equ     10      ; invisible file in dir command

                   ;       utility functions for file access

                   dm$position:
                           ; Compute disk map position for vrecord to hl
0962 21A71C4E              lxi h,blkshf! mov c,m ; shift count to c
0966 3ACA1C                lda vrecord ; current virtual record to a
                           dmpos0:
0969 B71F0DC269                    ora a! rar! dcr c! jnz dmpos0
                           ; a = shr(vrecord,blkshf) = vrecord/2**(sect/block)
096F 47                    mov b,a ; Save it for later addition
0970 3E0896                mvi a,8! sub m ; 8-blkshf to accumulator
0973 4F                    mov c,a ; extent shift count in register c
0974 3AC81C                lda extval ; extent value ani extmsk
```

```
                      dmpos1:
                              ; blkshf = 3,4,5,6,7, c=5,4,3,2,1
                              ; shift is 4,3,2,1,0
0977 0DCA8009                 dcr c! jz dmpos2
097B B717C37709               ora a! ral! jmp dmpos1
                      dmpos2:
                      ; Arrive here with a = shl(ext and extmsk,7-blkshf)
0980 80               add b ; Add the previous shr(vrecord,blkshf) value
                      ; a is one of the following values, depending upon alloc
                      ; bks blkshf
                      ; 1K   3      v/8 + extval * 16
                      ; 2K   4      v/16+ extval * 8
                      ; 4K   5      v/32+ extval * 4
                      ; 8K   6      v/64+ extval * 2
                      ; 16K  7      v/128+extval * 1
0981 C9               ret ; with dm$position in a

               getdma:
0982 2ADB1E1110       lhld info! lxi d,dskmap! dad d! ret

               getdm:
                      ; Return disk map value from position given by bc
098A CD8209           call getdma
098D 09               dad b ; Index by a single byte value
098E 3AC51C           lda single ; single byte/map entry?
0991 B7CA9809         ora a! jz getdmd ; Get disk map single byte
0995 6E60C9                   mov l,m! mov h,b! ret ; with hl=00bb
                      getdmd:
0998 09                       dad b ; hl=.fcb(dm+i*2)
                              ; double precision value returned
0999 7E23666FC9               mov a,m! inx h! mov h,m! mov l,a! ret

               index:
                      ; Compute disk block number from current fcb
099E CD6209           call dm$position ; 0...15 in register a
09A1 32C41C           sta dminx
09A4 4F0600CD8A       mov c,a! mvi b,0! call getdm ; value to hl
09AA 22CD1C7DB4       shld arecord! mov a,l! ora h! ret

               atran:
                      ; Compute actual record address, assuming index called

               ;      arecord = shl(arecord,blkshf)

09B0 3AA71C4F         lda blkshf! mov c,a
09B4 2ACD1CAFCD       lhld arecord! xra a! call shl3bv
09BB 22CD1C32CF       shld arecord! sta arecord+2

09C1 22D31C           shld arecord1 ; Save low(arecord)

               ;      arecord = arecord or (vrecord and blkmsk)

09C4 3AA81C4F3A       lda blkmsk! mov c,a! lda vrecord! ana c
09CC 47               mov b,a ; Save vrecord & blkmsk in reg b & blk$off
09CD 32B81C           sta blk$off
09D0 21CD1CB677       lxi h,arecord! ora m! mov m,a! ret
```

```

                         get$atts:
                                 ; Get volatile attributes starting at f'5
                                 ; info locates fcb
 09D6 2ADB1E                     lhld info
 09D9 11080019                   lxi d,8! dad d ; hl = .fcb(f'8)
 09DD 0E04                       mvi c,4
                         get$atts$loop:
 09DF 7E87F5                     mov a,m! add a! push a
 09E2 7A1F57                     mov a,d! rar! mov d,a
 09E5 F10F77                     pop a! rrc! mov m,a
 09E8 2B0DC2DF09                 dcx h! dcr c! jnz get$atts$loop
 09ED 7AC9                       mov a,d! ret

                         get$s1:
                                 ; Get current s1 field to a
 09EF CDFD09237E                 call getexta! inx h! mov a,m! ret

                         get$rra:
                                 ; Get current ran rec field address to hl
 09F5 2ADB1E1121                 lhld info! lxi d,ranrec! dad d ; hl=.fcb(ranrec)
 09FC C9                         ret

                         getexta:
                                 ; Get current extent field address to hl
 09FD 2ADB1E110C                 lhld info! lxi d,extnum! dad d ; hl=.fcb(extnum)
 0A04 C9                         ret

                         getrcnta:
                                 ; Get reccnt address to hl
 0A05 2ADB1E110F                 lhld info! lxi d,reccnt! dad d! ret

                         getfcba:
                                 ; Compute reccnt and nxtrec addresses for get/setfcb
 0A0D CD050AEB                   call getrcnta! xchg ; de=.fcb(reccnt)
 0A11 21110019                   lxi h,(nxtrec-reccnt)! dad d ; hl=.fcb(nxtrec)
 0A15 C9                         ret

                         getfcb:
                                 ; Set variables from currently addressed fcb
 0A16 CD0D0A                     call getfcba ; addresses in de, hl
 0A19 7E32CA1C                   mov a,m! sta vrecord ; vrecord=fcb(nxtrec)
 0A1D EB7EB7C22B                 xchg! mov a,m! ora a! jnz getfcb0
 0A23 CD2D0E4FCD                 call get$dir$ext! mov c,a! call set$rc! mov a,m
                         getfcb0:
 0A2B FE81DA320A                 cpi 81h! jc getfcb1
 0A30 3E80                       mvi a,80h
                         getfcb1:
 0A32 32C71C                     sta rcount ; rcount=fcb(reccnt) or 80h
 0A35 CDFD09                     call getexta ; hl=.fcb(extnum)
 0A38 3AA91C                     lda extmsk ; extent mask to a
 0A3B A6                         ana m ; fcb(extnum) and extmsk
 0A3C 32C81C                     sta extval
 0A3F C9                         ret

                         setfcb:
                                ; Place values back into current fcb
0A40 CD0D0A                     call getfcba ; addresses to de, hl
                                ; fcb(cr) = vrecord
0A43 3ACA1C77                   lda vrecord! mov m,a
                                ; Is fx < 22? (sequential read or write)
0A47 3ADF1EFE16                 lda fx! cpi 22! jnc $+4 ; no
                                ; fcb(cr) = fcb(cr) + 1
0A4F 34                         inr m
0A50 EB7EFE80D0                 xchg! mov a,m! cpi 80h! rnc ; dont reset fcb(rc) if > 7fh
0A55 3AC71C77                   lda rcount! mov m,a ; fcb(reccnt)=rcount
0A59 C9                         ret

                      zero$ext$mod:
0A5A CDFD097223                 call getexta! mov m,d! inx h! inx h! mov m,d
0A61 C9                         ret

                      zero:
0A62 70230DC8                   mov m,b! inx h! dcr c! rz
0A66 C3620A                     jmp zero

                      hlrotr:
                                ; hl rotate right by amount c
0A69 0C                         inr c ; in case zero
0A6A 0DC8                       hlrotr0: dcr c! rz ; return when zero
0A6C 7CB71F67                           mov a,h! ora a! rar! mov h,a ; high byte
0A70 7D1F6F                             mov a,l! rar! mov l,a ; low byte
0A73 C36A0A                             jmp hlrotr0

                      compute$cs:
                                ; Compute checksum for current directory buffer
0A76 2A8C1C                     lhld buffa ; current directory buffer
0A79 010400                     lxi b,4 ; b = 0, c = 4
                      compute$cs0:
0A7C 1620                       mvi d,32 ; size of fcb
0A7E AF                         xra a ; clear checksum value
                      compute$cs1:
0A7F 862315                             add m! inx h! dcr d
0A82 C27F0A                             jnz compute$cs1
0A85 A8470D                             xra b! mov b,a! dcr c
0A88 C27C0A                             jnz compute$cs0
0A8B C9                         ret ; with checksum in a

                      if MPM

                      compute$cs:
                                ; Compute checksum for current directory buffer
                                mvi c,recsiz ; size of directory buffer
                                lhld buffa ; current directory buffer
                                xra a ; Clear checksum value
                                computecs0:
                                        add m! inx h! dcr c ; cs = cs+buff(recsiz-c)
                                        jnz computecs0
                                ret ; with checksum in a

                      chksum$fcb: ; Compute checksum for fcb
                                ; Add 1st 12 bytes of fcb + curdsk +
                ;      high$ext + xfcb$read$only + bbh
                lxi h,pdcnt! mov a,m
                inx h! add m ; Add high$ext
                inx h! add m ; Add xfcb$read$only
                inx h! add m ; Add curdsk
                adi 0bbh ; Add 0bbh to bias checksum
                lhld info! mvi c,12! call computecs0
                ; Skip extnum
                inx h
                ; Add fcb(s1)
                add m! inx h
                ; Skip modnum
                inx h
                ; Skip fcb(reccnt)
                ; Add disk map
                inx h! mvi c,16! call computecs0
                ora a! ret ; Z flag set if checksum valid

        set$chksum$fcb:
                call chksum$fcb! rz
                mov b,a! call gets1
                cma! add b! cma
                mov m,a! ret

        reset$chksum$fcb:
                xra a! sta comp$fcb$cks
                call chksum$fcb! rnz
                call get$s1! inr m! ret

        endif

        check$fcb:

        if MPM
                xra a! sta check$fcb4
        check$fcb1:
                call chek$fcb! rz
        check$fcb2:

                ani 0fh! jnz check$fcb3
                lda pdcnt! ora a! jz check$fcb3
                call set$sdcnt! sta dont$close
                call close1
                lxi h,lret! inr m! jz check$fcb3
                mvi m,0! call pack$sdcnt! mvi b,5
                call search$olist! rz
        check$fcb3:

                pop h ; Discard return address
        check$fcb4:
                nop
                mvi a,10! jmp sta$ret

        set$fcb$cks$flag:
                mvi a,0ffh! sta comp$fcb$cks! ret

```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
                else
0A8C CDEF092A96         call getsl! lhld lsn$add
0A92 BEC4A10A           cmp m! cnz chk$media$fcb
                endif

                chek$fcb:
0A96 3AD71C             lda high$ext

                if MPM

                        ; if ext & 0110$0000b = 0110$0000b then
                        ; set fcb(0) to 0 (user 0)

                        cpi 0110$0000b! jnz chek$fcb1
                else
0A99 B7C8               ora a! rz
                endif

0A9B 2ADB1EAF77         lhld info! xra a! mov m,a ; fcb(0) = 0
                chek$fcb1:

                if MPM
                        jmp chksum$fcb ; ret
                else
0AA0 C9                 ret

                chk$media$fcb:
                        ; fcb(s1) ~= DPH login sequence # field
                        ; Is fcb addr < bdosadd?

                if banked
                        lhld user$info
                else
0AA1 2ADB1E             lhld info
                endif

0AA4 EB2AFE1ECD         xchg! lhld bdosadd! call subdh! jnc chk$media1 ; no
                        ; Is rlog(drive) true?
0AAE 2A921ECDDB         lhld rlog! call testvector! rz ; no
                chk$media1:
                        ; Return invalid fcb error code
0AB5 E1E1               pop h! pop h
                chk$media2:
0AB7 3E0AC3CB06         mvi a,10! jmp sta$ret
                endif

                hlrotl:
                        ; Rotate the mask in hl by amount in c
0ABC 0C                 inr c ; may be zero
0ABD 0DC8               hlrotl0: dcr c! rz ; return if zero
0ABF 29C3BD0A                   dad h! jmp hlrotl0

                set$dlog:
0AC3 11881C             lxi d,dlog
                set$cdisk:
                        ; Set a '1' value in curdsk position of bc
```

```
0AC6 3ACB1C             lda curdsk
                set$cdisk1:
0AC9 4F                 mov c,a ; Ready parameter for shift
0ACA 210100             lxi h,1 ; number to shift
0ACD CDBC0A             call hlrotl ; hl = mask to integrate
0AD0 1AB51213           ldax d! ora l! stax d! inx d
0AD4 1AB412C9           ldax d! ora h! stax d! ret

                nowrite:
                        ; Return true if dir checksum difference occurred
0AD8 2A861C             lhld rodsk

                test$vector:
0ADB 3ACB1C             lda curdsk
                test$vector1:
0ADE 4FCD690A           mov c,a! call hlrotr
0AE2 7DE601C9           mov a,l! ani 1b! ret ; non zero if curdsk bit on

                check$rodir:
                        ; Check current directory element for read/only status
0AE6 CDFE0A             call getdptra ; address of element

                check$rofile:
                        ; Check current buff(dptr) or fcb(0) for r/o status
0AE9 CDF00A             call ro$test
0AEC D0                 rnc ; Return if not set
0AED C38107             jmp rof$error ; Exit to read only disk message

                ro$test:
0AF0 11090019           lxi d,rofile! dad d
0AF4 7E17C9             mov a,m! ral! ret ; carry set if r/o

                check$write:
                        ; Check for write protected disk
0AF7 CDD80AC8           call nowrite! rz ; ok to write if not rodsk
0AFB C37C07             jmp rod$error ; read only disk error

                getdptra:
                        ; Compute the address of a directory element at
                        ; positon dptr in the buffer

0AFE 2A8C1C3AD8         lhld buffa! lda dptr
                addh:
                        ; hl = hl + a
0B04 856FD0             add l! mov l,a! rnc
                        ; overflow to h
0B07 24C9               inr h! ret

                getmodnum:
                        ; Compute the address of the module number
                        ; bring module number to accumulator
                        ; (high order bit is fwf (file write flag)
0B09 2ADB1E110E         lhld info! lxi d,modnum! dad d ; hl=.fcb(modnum)
0B10 7EC9               mov a,m! ret ; a=fcb(modnum)

                clrmodnum:
                        ; Clear the module number field for user open/make
0B12 CD090B3600         call getmodnum! mvi m,0 ; fcb(modnum)=0
0B17 C9                 ret

                clr$ext:
                        ; fcb ext = fcb ext & 1fh
0B18 CDFD097EE6         call getexta! mov a,m! ani 0001$1111b! mov m,a!
0B1F C9                 ret

                setfwf:
0B20 CD090B             call getmodnum ; hl=.fcb(modnum), a=fcb(modnum)
                        ; Set fwf (file write flag) to '1'
0B23 F68077             ori fwfmsk! mov m,a ; fcb(modnum)=fcb(modnum) or 80h
                        ; also returns non zero in accumulator
0B26 C9                 ret

                compcdr:
                        ; Return cy if cdrmax > dcnt
0B27 2AE11EEB           lhld dcnt! xchg ; de = directory counter
0B2B 2A8E1C             lhld cdrmaxa ; hl=.cdrmax
0B2E 7B96               mov a,e! sub m ; low(dcnt) - low(cdrmax)
0B30 23                 inx h ; hl = .cdrmax+1
0B31 7A9E               mov a,d! sbb m ; hig(dcnt) - hig(cdrmax)
                        ; condition dcnt - cdrmax  produces cy if cdrmax>dcnt
0B33 C9                 ret

                setcdr:
                        ; if not (cdrmax > dcnt) then cdrmax = dcnt+1
0B34 CD270B             call compcdr
0B37 D8                 rc ; Return if cdrmax > dcnt
                        ; otherwise, hl = .cdrmax+1, de = dcnt
0B38 13722B73           inx d! mov m,d! dcx h! mov m,e
0B3C C9                 ret

                subdh:
                        ; Compute hl = de - hl
0B3D 7B956F7A9C         mov a,e! sub l! mov l,a! mov a,d! sbb h! mov h,a
0B43 C9                 ret

                newchecksum:
0B44 0EFE               mvi c,0feh ; Drop through to compute new checksum
                checksum:
                        ; Compute current checksum record and update the
                        ; directory element if c=true, or check for = if not
                        ; drec < chksiz?
0B46 2AB61CEB2A         lhld drec! xchg! lhld chksiz
0B4D 7CE67F67           mov a,h! ani 7fh! mov h,a ; Mask off permanent drive bit
0B51 CD3D0B             call subdh ; de-hl
0B54 D0                 rnc ; Skip checksum if past checksum vector size
                                ; drec < chksiz, so continue
0B55 C5                         push b ; Save init flag
0B56 CD760A                     call compute$cs ; Check sum value to a
0B59 2A9A1C                     lhld checka ; address of check sum vector
0B5C EB                         xchg
0B5D 2AB61C                     lhld drec
0B60 19                         dad d ; hl = .check(drec)
```

```
0B61 C1                       pop b ; Recall true=0ffh or false=00 to c
0B62 0C                       inr c ; 0ffh produces zero flag
0B63 CA840B                   jz initial$cs
0B66 0C                       inr c ; 0feh produces zero flag
0B67 CA8F0B                   jz update$cs

               if MPM
                              inr c! jz test$dir$cs
               endif

                                      ; not initializing, compare
0B6A BE                               cmp m ; compute$cs=check(drec)?
0B6B C8                               rz ; no message if ok
                                      ; checksum error, are we beyond
                                      ; the end of the disk?
0B6C CDD80AC0                         call nowrite! rnz

               media$change:
0B70 CD100C                           call discard$data

               if MPM
                                      call flush$file0
               else
0B73 3EFF32DE1E                       mvi a,0ffh! sta relog! sta hashl
0B7B CDA60B                           call set$rlog
               endif

                                      ; Reset the drive

0B7E CDC30AC3F4                       call set$dlog! jmp reset37x

               if MPM
                              test$dir$cs:
                                      cmp m! jnz flush$files
                                      ret
               endif

                              initial$cs:
                                      ; initializing the checksum
0B84 BE77C8                           cmp m! mov m,a! rz
                                      ; or 1 into login seq # if media change
0B87 2A961C3E01                       lhld lsn$add! mvi a,1! ora m! mov m,a! ret

                              update$cs:
                                      ; updating the checksum
0B8F 77C9                             mov m,a! ret

               set$ro:
                      ; Set current disk to read/only
0B91 3AC61C1186       lda seldsk! lxi d,rodsk! call set$cdisk1 ; sets bit to 1
                      ; high water mark in directory goes to max
0B9A 2AAC1C23EB       lhld dirmax! inx h! xchg ; de = directory max
0B9F 2A8E1C           lhld cdrmaxa ; hl = .cdrmax
0BA2 732372           mov m,e! inx h! mov m,d ; cdrmax = dirmax
0BA5 C9               ret

```

```
                        set$rlog:
                                ; rlog(seldsk) = true
0BA6 2A901ECDDB                 lhld olog! call test$vector! rz
0BAD 11921EC3C6                 lxi d,rlog! jmp set$cdisk

                        tst$log$fxs:
0BB3 3AB11CE680                 lda chksiz+1! ani 80h! rnz
0BB9 21E41C                     lxi h,log$fxs
                        tst$log0:
0BBC 3ADF1E47                   lda fx! mov b,a
                        tst$log1:
0BC0 7EB8C8                     mov a,m! cmp b! rz
0BC3 23B7C2C00B                 inx h! ora a! jnz tst$log1
0BC8 3CC9                       inr a! ret

                        test$media$flag:
0BCA 2A961C237E                 lhld lsn$add! inx h! mov a,m! ora a! ret

                        chk$exit$fxs:
0BD1 21641CE5                   lxi h,goback! push h
                                ; does fx = read or write function?
                                ; and is drive removable?
0BD5 21F11CCDBC                 lxi h,rw$fxs! call tst$log0! jz chk$media2 ; yes
                                ; is fx = close or searchn function?
                                ; and is drive removable?
0BDE 21F81CCDBC                 lxi h,sc$fxs! call tst$log0! jz lret$eq$ff ; yes
0BE7 E1C9                       pop h! ret

                        tst$relog:
0BE9 21DE1E7EB7                 lxi h,relog! mov a,m! ora a! rz
0BEF 3600                       mvi m,0
                        drv$relog:
0BF1 CD8016                     call curselect
0BF4 21000022E1                 lxi h,0! shld dcnt! xra a! sta dptr
0BFE C9                         ret

                        set$lsn:
0BFF 2A961C4E                   lhld lsn$add! mov c,m
0C03 CDEF0971C9                 call getsl! mov m,c! ret

                        discard$data$bcb:
0C08 2AA01C0E04                 lhld dtabcba! mvi c,4! jmp discard0

                        discard$data:
0C10 2AA01CC319                 lhld dtabcba! jmp discard

                        discard$dir:
0C16 2A9E1C                     lhld dirbcba

                        discard:
0C19 0E01                       mvi c,1
                        discard0:
0C1B 7DA43CC8                   mov a,l! ana h! inr a! rz

                        if BANKED
                                mov e,m! inx h! mov d,m! xchg
```

```
                    discard1:
                            push h! push b
                            lxi d,adrive! call compare
                            pop b! pop h! jnz discard2

                            mvi m,0ffh
                    discard2:
                            lxi d,13! dad d
                            mov e,m! inx h! mov d,m
                            xchg! mov a,l! ora h! rz
                            jmp discard1
                    else
0C1F E5                     push h
0C20 11CC1CCDEB             lxi d,adrive! call compare
0C26 E1C0                   pop h! rnz
0C28 36FFC9                 mvi m,0ffh! ret
                    endif

                    get$buffa:
0C2B D5110A0019             push d! lxi d,10! dad d
0C30 5E2356                 mov e,m! inx h! mov d,m

                    if BANKED
                            inx h! mov a,m! sta buffer$bank
                    endif

0C33 EBD1C9                 xchg! pop d! ret

                    rddir:
                            ; Read a directory entry into the directory buffer
0C36 CDB308                 call seek$dir
0C39 3E03C3460C             mvi a,3! jmp wrdir0

                    seek$copy:
                    wrdir:
                            ; Write the current directory entry, set checksum
0C3E CDF70A                 call check$write
0C41 CD440B                 call newchecksum ; Initialize entry
0C44 3E05                   mvi a,5
                    wrdir0:
0C46 21000022BB             lxi h,0! shld last$block
0C4C 2A9E1C                 lhld dirbcba

                    if BANKED
                            cpi 5! jnz $+6
                            lhld curbcba
                    endif

0C4F CD051D                 call deblock

                    setdata:
                            ; Set data dma address
0C52 2AD81EC35B             lhld dmaad! jmp setdma ; to complete the call

                    setdir1:
0C58 CD2B0C                 call get$buffa
```

```

                      setdma:
                              ; hl=.dma address to set (i.e., buffa or dmaad)
0C5B 228A1CC9                 shld curdma! ret

                      dir$to$user:

                      if not MPM
                              ; Copy the directory entry to the user buffer
                              ; after call to search or searchn by user code
0C5F 2A8C1CEB                 lhld buffa! xchg ; source is directory buffer
0C63 2A8A1C                   lhld xdmaad ; destination is user dma address
0C66 018000                   lxi b,recsiz ; copy entire record
0C69 CD4BFF                   call movef
                      endif
                              ; Set lret to dcnt & 3 if search successful
0C6C 217A077E3C               lxi h,lret! mov a,m! inr a! rz
0C72 3AE11EE603               lda dcnt! ani dskmsk! mov m,a! ret

                      make$fcb$inv: ; Flag fcb as invalid
                              ; Reset fcb write flag
0C79 CD200B                   call setfwf
                              ; Set 1st two bytes of diskmap to ffh
0C7C 23233EFF77               inx h! inx h! mvi a,0ffh! mov m,a! inx h! mov m,a
0C83 C9                       ret

                      chk$inv$fcb: ; Check for invalid fcb
0C84 CD8209C397               call getdma! jmp test$ffff

                      tst$inv$fcb: ; Test for invalid fcb
0C8A CD840CC0                 call chk$inv$fcb! rnz
0C8E E13E09C3CB               pop h! mvi a,9! jmp sta$ret! ; lret = 9

                      end$of$dir:
                              ; Return zero flag if at end of directory, non zero
                              ; if not at end (end of dir if dcnt = 0ffffh)
0C94 21E11E                   lxi h,dcnt
                      test$ffff:
0C97 7E                       mov a,m ; may be 0ffh
0C98 23BE                     inx h! cmp m ; low(dcnt) = high(dcnt)?
0C9A C0                       rnz ; non zero returned if different
                              ; high and low the same, = 0ffh?
0C9B 3C                       inr a ; 0ffh becomes 00 if so
0C9C C9                       ret

                      set$end$dir:
                              ; Set dcnt to the end of the directory
0C9D 21FFFF22E1               lxi h,enddir! shld dcnt! ret

                      read$dir:
0CA4 CDAA0CC3DC               call r$dir! jmp r$dir1

                      r$dir:
                              ; Read next directory entry, with c=true if initializing

0CAA 2AAC1CEB                 lhld dirmax! xchg ; in preparation for subtract
```

```
0CAE 2AE11E2322       lhld dcnt! inx h! shld dcnt ; dcnt=dcnt+1
                      ; Continue while dirmax >= dcnt (dirmax-dcnt no cy)
0CB5 CD3D0B           call subdh ; de-hl

0CB8 DA9D0C           jc set$end$dir

                      read$dir0:
                              ; not at end of directory, seek next element
                              ; initialization flag is in c
0CBB 3AE11EE603               lda dcnt! ani dskmsk ; low(dcnt) and dskmsk
0CC0 0605                     mvi b,fcbshf ; to multiply by fcb size
                              read$dir1:
                                      add a! dcr b! jnz read$dir1
                              ; a = (low(dcnt) and dskmsk) shl fcbshf
0CC7 32D81C                   sta dptr ; ready for next dir operation
0CCA B7C0                     ora a! rnz ; Return if not a new record
                      read$dir2:
0CCC C5                       push b ; Save initialization flag c
0CCD CD360C                   call rd$dir ; Read the directory record
0CD0 C1                       pop b ; Recall initialization flag
0CD1 3ADE1EB7C0               lda relog! ora a! rnz
0CD6 C34608                   jmp checksum ; Checksum the directory elt

               r$dir2:
0CD9 CDCC0C           call read$dir2
               r$dir1:
0CDC 3ADE1EB7C8       lda relog! ora a! rz
0CE1 CDD10B           call chk$exit$fxs
0CE4 CDE90BC336       call tst$relog! jmp rd$dir

               getallocbit:
                      ; Given allocation vector position bc, return with byte
                      ; containing bc shifted so that the least significant
                      ; bit is in the low order accumulator position.  hl is
                      ; the address of the byte for possible replacement in
                      ; memory upon return, and d contains the number of shifts
                      ; required to place the returned value back into position
0CEA 79E6073C5F       mov a,c! ani 111b! inr a! mov e,a! mov d,a
                      ; d and e both contain the number of bit positions to shift

0CF0 60690E03         mov h,b! mov l,c! mvi c,3 ; bc = bc shr 3
0CF4 CD690A           call hlrotr ; hlrotr does not touch d and e
0CF7 444D             mov b,h! mov c,l

0CF9 2A9C1C           lhld alloca ; base address of allocation vector
0CFC 097E             dad b! mov a,m ; byte to a, hl = .alloc(bc shr 3)
                      ; Now move the bit to the low order position of a
0CFE 071DC2FE0C       rotl: rlc! dcr e! jnz rotl! ret

               setallocbit:
                      ; bc is the bit position of alloc to set or reset.  the
                      ; value of the bit is in register e.
0D04 D5CDEA0C         push d! call getallocbit ; shifted val a, count in d
0D08 E6FE             ani 1111$1110b ; mask low bit to zero (may be set)
0D0A C1B1             pop b! ora c ; low bit of c is masked into a
                      ; jmp rotr ; to rotate back into proper position
```

```
                       ; ret

                rotr:
                       ; byte value from alloc is in register a, with shift count
                       ; in register c (to place bit back into position), and
                       ; target alloc position in registers hl, rotate and replace
0D0C 0F15C20C0D        rrc! dcr d! jnz rotr ; back into position
0D11 77                mov m,a ; back to alloc
0D12 C9                ret

                copy$alv:
                       ; If Z flag set, copy 1st ALV to 2nd
                       ; Otherwise, copy 2nd ALV to 1st

                if not BANKED
0D13 3AF31E0707        lda bdos$flags! rlc! rlc! rc
                endif

0D19 F5                push a
0D1A CD860D444D        call get$nalbs! mov b,h! mov c,l
0D1F 2A9C1C545D        lhld alloca! mov d,h! mov e,l! dad b
0D25 F1CA4BFF          pop a! jz movef
0D29 EBC34BFF          xchg! jmp movef

                scandm$ab:
                       ; Set/Reset 1st and 2nd ALV
0D2D C5CD4E0D          push b! call scandm$a
0D31 C1                pop b! ;jmp scandm$b

                scandm$b:
                       ; Set/Reset 2nd ALV

                if not BANKED
0D32 3AF31EE640        lda bdos$flags! ani 40h! rnz
                endif

0D38 C5CD860D          push b! call get$nalbs
0D3C EB2A9C1C          xchg! lhld alloca
0D40 C1E519229C        pop b! push h! dad d! shld alloca
0D46 CD4E0D            call scandm$a
0D49 E1229C1CC9        pop h! shld alloca! ret

                scandm$a:
                       ; Set/Reset 1st ALV
                       ; Scan the disk map addressed by dptr for non-zero
                       ; entries, the allocation vector entry corresponding
                       ; to a non-zero entry is set to the value of c (0,1)
0D4E CDFE0A            call getdptra ; hl = buffa + dptr
                       ; hl addresses the beginning of the directory entry
0D51 11100019          lxi d,dskmap! dad d ; hl now addresses the disk map
0D55 C5                push b ; Save the 0/1 bit to set
0D56 0E11              mvi c,fcblen-dskmap+1 ; size of single byte disk map + 1
                       scandm0:
                               ; Loop once for each disk map entry
0D58 D1                        pop d ; Recall bit parity
0D59 0DC8                      dcr c! rz ; all done scanning?
```

```
                                    ; no, get next entry for scan
0D5B D5                             push d ; Replace bit parity
0D5C 3AC51CB7CA                     lda single! ora a! jz scandm1
                                            ; single byte scan operation
0D63 C5                                     push b ; Save counter
0D64 E5                                     push h ; Save map address
0D65 4E0600                                 mov c,m! mvi b,0 ; bc=block#
0D68 C3710D                                 jmp scandm2
                                    scandm1:
                                            ; double byte scan operation
0D6B 0D                                     dcr c ; count for double byte
0D6C C5                                     push b ; Save counter
0D6D 4E2346                                 mov c,m! inx h! mov b,m ; bc=block#
0D70 E5                                     push h ; Save map address
                                    scandm2:
                                            ; Arrive here with bc=block#, e=0/1
0D71 79B0                                   mov a,c! ora b ; Skip if = 0000
0D73 CA800D                                 jz scandm3
0D76 2AAA1C                                 lhld maxall ; Check invalid index
0D79 7D917C98                               mov a,l! sub c! mov a,h! sbb b ; maxall - block#
0D7D D4040D                                 cnc set$alloc$bit
                                            ; bit set to 0/1
                                    scandm3:
0D80 E123                                   pop h! inx h ; to next bit position
0D82 C1                                     pop b ; Recall counter
0D83 C3580D                                 jmp scandm0 ; for another item

                        get$nalbs: ; Get # of allocation vector bytes
0D86 2AAA1C0E03                 lhld maxall! mvi c,3
                                ; number of bytes in allocation vector is (maxall/8)+1
0D8B CD690A23C9                 call hlrotr! inx h! ret

                        if MPM

                        test$dir:
                                call home
                                call set$end$dir
                        test$dir1:
                                mvi c,0feh! call read$dir
                                lda flushed! ora a! rnz
                                call end$of$dir! rz
                                jmp test$dir1
                        endif

                        initialize:
                                ; Initialize the current disk
                                ; lret = false ; set to true if $ file exists
                                ; Compute the length of the allocation vector - 2

                        if MPM
                                lhld tlog! call test$vector! jz initialize1
                                lhld tlog! call remove$drive! shld tlog
                                xra a! sta flushed
                                call test$dir! rz
                        initialize1:
                        else
```

```
0D90 CDCA0B3600       call test$media$flag! mvi m,0 ; Reset media change flag
0D95 CD100C           call discard$data
0D98 CD160C           call discard$dir
               endif

               if BANKED
                      ; Is drive permanent with no chksum vector?
                      call chksiz$eq$8000h! jnz initialize2 ; no
                      ; Is this an initial login operation?
                      ; register A = 0
                      lhld lsn$add! cmp m! mvi m,2! jz initialize2 ; yes
                      jmp copy$alv ; Copy 2nd ALV to 1st ALV
               initialize2:

               endif

0D9B CD860D           call get$nalbs ; Get # of allocation vector bytes
0D9E 444D             mov b,h! mov c,l ; Count down bc til zero
0DA0 2A9C1C           lhld alloca ; base of allocation vector
                      ; Fill the allocation vector with zeros
                      initial0:
0DA3 360023                   mvi m,0! inx h ; alloc(i)=0
0DA6 0B                       dcx b ; Count length down
0DA7 78B1C2A30D               mov a,b! ora c! jnz initial0

0DAC 2A941C77         lhld drvlbla! mov m,a ; Zero out drive desc byte

                      ; Set the reserved space for the directory

0DB0 2AAE1CEB         lhld dirblk! xchg
0DB4 2A9C1C           lhld alloca ; hl=.alloc()
0DB7 732372           mov m,e! inx h! mov m,d ; sets reserved directory blks
                      ; allocation vector initialized, home disk
0DBA CD4C08           call home
                       ; cdrmax = 3 (scans at least one directory record)
0DBD 2A8E1C3604       lhld cdrmaxa! mvi m,4! inx h! mvi m,0

0DC5 CD9D0C           call set$end$dir ; dcnt = enddir
0DC8 2AA21C22D3       lhld hashtbla! shld arecord1

                      ; Read directory entries and check for allocated storage

                      initial2:
0DCE 0EFFCDA40C               mvi c,true! call read$dir
0DD3 CD940CCA13               call end$of$dir! jz copy$alv
                              ; not end of directory, valid entry?
0DD9 CDFE0A                   call getdptra ; hl = buffa + dptr
0DDC EB2AD31C7C               xchg! lhld arecord1! mov a,h! ana l! inr a! xchg
                              ; is hashtbla ^= 0ffffh
0DE4 C4001F                   cnz init$hash ; yes - call init$hash
0DE7 3E21BE                   mvi a,21h! cmp m
0DEA CACE0D                   jz initial2 ; Skip date & time records

0DED 3EE5BE                   mvi a,empty! cmp m
0DF0 CACE0D                   jz initial2 ; go get another item

```

```
0DF3 3E20BECA0A                 mvi a,20h! cmp m! jz drv$lbl
0DF9 3E10A6C204                 mvi a,10h! ana m! jnz initial3

                                ; Now scan the disk map for allocated blocks

0DFF 0E01                       mvi c,1 ; set to allocated
0E01 CD4E0D                     call scandm$a
                        initial3:
0E04 CD340B                     call setcdr ; set cdrmax to dcnt
0E07 C3CE0D                     jmp initial2 ; for another entry

                drv$lbl:
0E0A 110C00197E                 lxi d,extnum! dad d! mov a,m
0E0F 2A941C77C3                 lhld drvlbla! mov m,a! jmp initial3

                copy$dirloc:
                        ; Copy directory location to lret following
                        ; delete, rename, ... ops

0E16 3AC21CC3CB         lda dirloc! jmp sta$ret
                        ; ret

                compext:
                        ; Compare extent# in a with that in c, return nonzero
                        ; if they do not match
0E1C C5                 push b ; Save c's original value
0E1D F53AA91C2F         push psw! lda extmsk! cma! mov b,a
                        ; b has negated form of extent mask
0E23 79A04F             mov a,c! ana b! mov c,a ; low bits removed from c
0E26 F1A0               pop psw! ana b ; low bits removed from a
0E28 91E61F             sub c! ani maxext ; Set flags
0E2B C1                 pop b ; Restore original values
0E2C C9                 ret

                get$dir$ext:
                        ; Compute directory extent from fcb
                        ; Scan fcb disk map backwards
0E2D CD0D0A             call getfcba ; hl = .fcb(vrecord)
0E30 0E10410CC5         mvi c,16! mov b,c! inr c! push b
                        ; b=dskmap pos (rel to 0)
                get$de0:
0E35 C1                 pop b
0E36 0D                 dcr c
0E37 AF                 xra a ; Compare to zero
                get$de1:
0E38 2B05               dcx h! dcr b; Decr dskmap position
0E3A BEC2420E           cmp m! jnz get$de2 ; fcb(dskmap(b)) ^= 0
0E3E 0DC2380E           dcr c! jnz get$de1
                        ; c = 0 -> all blocks = 0 in fcb disk map
                get$de2:
0E42 7932C41C           mov a,c! sta dminx
0E46 3AC51CB778         lda single! ora a! mov a,b
0E4B C24F0E             jnz get$de3
0E4E 1F                 rar ; not single, divide blk idx by 2
                get$de3:
0E4F C5E5               push b! push h ; Save dskmap position & count
```

```
0E51 6F2600             mov l,a! mvi h,0 ; hl = non-zero blk idx
                        ; Compute ext offset from last non-zero
                        ; block index by shifting blk idx right
                        ; 7 - blkshf
0E54 3AA71C573E         lda blkshf! mov d,a! mvi a,7! sub d
0E5B 4FCD690A45         mov c,a! call hlrotr! mov b,l
                        ; b = ext offset
0E60 3AA91CB8E1         lda extmsk! cmp b! pop h! jc get$de0
                        ; Verify computed extent offset <= extmsk
0E68 CDFD094E           call getexta! mov c,m
0E6C 2FE61FA1B0         cma! ani maxext! ana c! ora b
                        ; dir ext = (fcb ext & (~ extmsk) & maxext) ! ext offset
0E71 C1                 pop b ; Restore stack
0E72 C9                 ret ; a = directory extent

                searchi:
                        ; search initialization
0E73 2ADB1E22E3         lhld info! shld searcha ; searcha = info
                searchil:
0E79 7932E51E           mov a,c! sta searchl ; searchl = c
0E7D CD141F             call set$hash
0E80 3EFF32C21C         mvi a,0ffh! sta dirloc ; changed if actually found
0E85 C9                 ret

                search$namlen:
0E86 0E0FC38D0E         mvi c,namlen! jmp search
                search$extnum:
0E8B 0E0C               mvi c,extnum
                search:
                        ; Search for directory element of length c at info
0E8D CD730E             call searchi
                searchl: ; entry point used by rename
0E90 CD9D0C             call set$end$dir ; dcnt = enddir
0E93 CDB30BCC4C         call tst$log$fxs! cz home
                        ; (drop through to searchn)

                searchn:
                        ; Search for the next directory element, assuming
                        ; a previous call on search which sets searcha and
                        ; searchl

                if MPM
                        lxi h,user0$pass! xra a! cmp m! mov m,a! cnz swap
                else
0E99 AF32E21C           xra a! sta user0$pass
                endif

0E9D CDD91FC220         call search$hash! jnz search$fin
0EA3 0E00CDA40C         mvi c,false! call read$dir ; Read next dir element
0EA8 CD940CCA20         call end$of$dir! jz search$fin
                                    ; not end of directory, scan for match
0EAE 2AE31EEB                       lhld searcha! xchg ; de=beginning of user fcb
0EB2 1A                             ldax d ; first character
0EB3 FEE5                           cpi empty ; Keep scanning if empty
0EB5 CAC00E                         jz searchnext
                                    ; not empty, may be end of logical directory
```

```
0EB8 D5                       push d ; Save search address
0EB9 CD270B                   call compcdr ; past logical end?
0EBC D1                       pop d ; Recall address
0EBD D2200F                   jnc search$fin ; artificial stop
               searchnext:
0EC0 CDFE0A                   call getdptra ; hl = buffa+dptr
0EC3 3AE51E4F                 lda searchl! mov c,a ; length of search to c
0EC7 0600                     mvi b,0 ; b counts up, c counts down

0EC9 7EFEE5CC53               mov a,m! cpi empty! cz save$dcnt$pos1

               if BANKED
                              xra a! sta save$xfcb
                              mov a,m! ani 1110$1111b! cmp m! jz search$loop
                              xchg! cmp m! xchg! jnz search$loop
                              lda find$xfcb! ora a! jz search$n
                              sta save$xfcb! jmp searchok
               endif

                              searchloop:
0ECF 79B7CA130F                       mov a,c! ora a! jz endsearch
0ED4 1AFE3FCA0C                       ldax d! cpi '?'! jz searchok ; ? in user fcb
                                      ; Scan next character if not ubytes
0EDA 78FE0DCA0C                       mov a,b! cpi ubytes! jz searchok
                                      ; not the ubytes field, extent field?
0EE0 FE0C                             cpi extnum ; may be extent field
0EE2 CAF40E                           jz searchext ; Skip to search extent
0EE5 FE0E1ACC1D                       cpi modnum! ldax d! cz searchmod
0EEB 96E67F                           sub m! ani 7fh ; Mask-out flags/extent modulus
0EEE C22D0F                           jnz searchnm ; Skip if not matched
0EF1 C30C0F                           jmp searchok ; matched character
                              searchext:
0EF4 1A                               ldax d
                                      ; Attempt an extent # match
0EF5 C5                               push b ; Save counters

               if MPM
                                      push h
                                      lhld sdcnt
                                      inr h! jnz dont$save
                                      lhld dcnt! shld sdcnt
                                      lhld dblk! shld sdblk
                              dont$save:
                                      pop h
               endif

0EF6 4E                               mov c,m ; directory character to c
0EF7 CD1C0E                           call compext ; Compare user/dir char

0EFA 47                               mov b,a
0EFB 3AE21C3CCA                       lda user0pass! inr a! jz save$dcnt$pos2
                                      ; Disable search of user 0 if any fcb
                                      ; is found under the current user #
0F02 AF32E11C                         xra a! sta search$user0
0F06 78                               mov a,b
```

```
0F07 C1                                         pop b ; Recall counters
0F08 B7                                         ora a ; Set flag
0F09 C2990E                                     jnz searchn ; Skip if no match
                                        searchok:
                                                ; current character matches
0F0C 1323040D                                   inx d! inx h! inr b! dcr c
0F10 C3CF0E                                     jmp searchloop
                                        endsearch:
                                                ; entire name matches, return dir position

                        if BANKED
                                                lda save$xfcb! inr a! jnz endsearch1
                                                lda xdcnt+1! cpi 0feh! cz save$dcnt$pos0
                                                jmp searchn
                                        endsearch1:
                        endif

0F13 AF32C21C                                   xra a! sta dirloc ; dirloc = 0
0F17 327A07                                     sta lret ; lret = 0
                                                ; successful search -
                                                ; return with zero flag reset
0F1A 4704C9                                     mov b,a! inr b! ret
                                        searchmod:
0F1D E63FC9                                     ani 3fh! ret ; Mask off high 2 bits
                                        search$fin:
                                                ; end of directory, or empty name

0F20 CD530F                                     call save$dcnt$pos1

                                                ; Set dcnt = 0ffffh
0F23 CD9D0C                                     call set$end$dir ; may be artifical end
                                        lret$eq$ff:
                                                ; unsuccessful search -
                                                ; return with zero flag set
                                                ; lret,low(aret) = 0ffh
0F26 3EFF4704C3                                 mvi a,255! mov b,a! inr b! jmp sta$ret

                                        searchnm: ; search no match routine
0F2D 78B7C2990E                                 mov a,b! ora a! jnz searchn ; fcb(0)?
0F32 7EB7C2990E                                 mov a,m! ora a! jnz searchn ; dir fcb(0)=0?
0F37 3AE11CB7CA                                 lda search$user0! ora a! jz searchn
0F3E 32E21C                                     sta user0$pass

                        if MPM
                                                call swap
                        endif

0F41 C30C0F                                     jmp searchok

                        if MPM

                        swap: ; Swap dcnt,sdblk with sdcnt0,sdblk0
                                push h! push d! push b
                                lxi d,sdcnt! lxi h,sdcnt0
                                mvi b,4
                        swap1:
                                ldax d! mov c,a! mov a,m
                                stax d! mov m,c
                                inx h! inx d! dcr b! jnz swap1
                                pop b! pop d! pop h!
                                ret
                        endif

                        save$dcnt$pos2:
                                ; Save directory position of matching fcb
                                ; under user 0 with matching extent # & modnum = 0
                                ; a = 0 on entry
0F44 B0C101990E                 ora b! pop b! lxi b,searchn! push b! rnz
0F4B 23237EB7C0                 inx h! inx h! mov a,m! ora a! rnz
                                ; Call if user0pass = 0ffh &
                                ;         dir fcb(extnum) = fcb(extnum)
                                ;         dir fcb(modnum) = 0
                        save$dcnt$pos0:
0F50 CD5B0F                     call save$dcnt$pos ; Return to searchn
                        save$dcnt$pos1:
                                ; Save directory position of first empty fcb
                                ; or the end of the directory
0F53 E5                         push h!
0F54 2ADF1C24C2                 lhld xdcnt! inr h! jnz save$dcnt$pos$ret ; Return if h ^= 0ffh
                        save$dcnt$pos:
0F5B 2AE11E22DF                 lhld dcnt! shld xdcnt

                        if MPM
                                lhld dblk! shld xdblk
                        endif

                        save$dcnt$pos$ret:
0F61 E1C9                       pop h! ret

                        if BANKED

                        init$xfcb$search:
                                mvi a,0ffh
                        init$xfcb$search1:
                                sta find$xfcb! mvi a,0feh! sta xdcnt+1! ret

                        does$xfcb$exist:
                                lda xdcnt+1! cpi 0feh! rz
                                call set$dcnt$dblk
                                xra a! call init$xfcb$search1
                                lhld searcha! mov a,m! ori 10h! mov m,a
                                mvi c,extnum! call searchi1! jmp searchn

                        xdcnt$eq$dcnt:
                                lhld dcnt! shld xdcnt! ret

                        restore$dir$fcb:
                                call set$dcnt$dblk
                                mvi c,namlen! call searchi! jmp searchn
                        endif

                        delete:
                              ; Delete the currently addressed file
0F63 CDD609                   call get$atts

                    if BANKED
                              sta attributes
                              ; Make search return matching fcbs and xfcbs
                    deletex:
                              mvi a,0feh! call init$xfcb$search1
                    else
                              ; Return with aret = 0 for XFCB only delete
                              ; in non-banked systems
0F66 17D8                     ral! rc
                    endif

                    ; Delete pass 1 - check r/o attributes and xfcb passwords

0F68 CD8B0EC8                 call search$extnum! rz

                        delete00:
0F6C CA840F                       jz delete1

                    if BANKED
                                  ; Is addressed dir fcb an xfcb?
                                  call getdptra! mov a,m! ani 10h! jnz delete01 ; yes

                    if MPM
                                  call tst$olist ; Verify fcb not open by someone else
                    endif

                                  ; Check r/o attribute if this is not an
                                  ; xfcb only delete operation.
                                  lda attributes! ral! cnc check$rodir
                    else
0F6F CDE60A                       call check$rodir
                    endif

                    if BANKED
                                  ; Are xfcb passwords enabled?
                                  call get$dir$mode! ral! jc delete02 ; no
                    endif

                                  ; Is this a wild card delete operation?
0F72 2ADB1ECD21                   lhld info! call chk$wild! jz delete02 ; yes
                                  ; Not wild & passwords inactive
                                  ; Skip to pass 2
0F7B C38A0F                       jmp delete11

                    if BANKED

                        delete01:
                                  ; Check xfcb password if passwords enabled
                                  call get$dir$mode! ral! jnc delete02
                                  call chk$xfcb$password! jz delete02
                                  call chk$pw$error! jmp deletex
                    endif

```

```
                          delete02:
0F7E CD990EC36C                 call searchn! jmp delete00

                      ; Delete pass 2 - delete all matching fcbs and/or xfcbs.

                      delete1:
0F84 CD8B0E                 call search$extnum

                          delete10:
0F87 CA160E                     jz copy$dir$loc
                          delete11:
0F8A CDFE0A                     call getdptra

                      if BANKED
                                ; Is addressed dir fcb an xfcb?
                                mov a,m! ani 10h! jnz delete12 ; yes
                      if MPM
                                push h
                                call chk$olist ; Delete olist item if present
                                pop h
                      endif
                                ; Is this delete operation xfcb only?
                                lda attributes! ani 80h! jnz delete13 ; yes
                      endif

                          delete12:
                                ; Delete dir fcb or xfcb
                                ; if fcb free all alocated blocks.

0F8D 36E5                       mvi m,empty

                      if BANKED

                          delete13:
                                push a ; Z flag set => free FCB blocks
                                ; Zero password mode byte in sfcb if sfcb exists
                                ; Does sfcb exist?
                                call get$dtba$8! ora a! jnz $+4 ; no
                                ; Zero mode byte
                                mov m,a
                      endif

0F8F CD3E0C0E00                 call wrdir! mvi c,0

                      if BANKED
                                pop a! cz scandm$ab
                      else
0F94 CD2D0D                     call scandm$ab
                      endif

0F97 CDB620                     call fix$hash
0F9A CD990EC387                 call searchn! jmp delete10

                      get$block:
                            ; Given allocation vector position bc, find the zero bit
                            ; closest to this position by searching left and right.
                          ; if found, set the bit to one and return the bit position
                          ; in hl.  if not found (i.e., we pass 0 on the left, or
                          ; maxall on the right), return 0000 in hl
0FA0 5059                 mov d,b! mov e,c ; copy of starting position to de
                          righttst:
0FA2 2AAA1C                       lhld maxall ; value of maximum allocation#
0FA5 7B957A9C                     mov a,e! sub l! mov a,d! sbb h ; right=maxall?
0FA9 D2D60F                       jnc retblock0 ; return block 0000 if so
0FAC 13C5D5                       inx d! push b! push d ; left, right pushed
0FAF 424B                         mov b,d! mov c,e ; ready right for call
0FB1 CDEA0C                       call getallocbit
0FB4 1FD2CE0F                     rar! jnc retblock ; Return block number if zero
0FB8 D1C1                         pop d! pop b ; Restore left and right pointers
                          lefttst:
0FBA 79B0CAA20F                   mov a,c! ora b! jz righttst ; Skip if left=0000
                                  ; left not at position zero, bit zero?
0FBF 0BD5C5                       dcx b! push d! push b ; left,right pushed
0FC2 CDEA0C                       call getallocbit
0FC5 1FD2CE0F                     rar! jnc retblock ; return block number if zero
                                  ; bit is one, so try the right
0FC9 C1D1                         pop b! pop d ; left, right restored
0FCB C3A20F                       jmp righttst
                          retblock:
0FCE 173C                         ral! inr a ; bit back into position and set to 1
                                  ; d contains the number of shifts required to reposition
0FD0 CD0C0D                       call rotr ; move bit back to position and store
0FD3 E1D1                         pop h! pop d ; hl returned value, de discarded
0FD5 C9                           ret
                          retblock0:
                                  ; cannot find an available bit, return 0000
0FD6 79                           mov a,c
0FD7 B0C2BA0F                     ora b! jnz lefttst ; also at beginning
0FDB 210000C9                     lxi h,0000h! ret

                  copy$dir:
                          ; Copy fcb information starting at c for e bytes
                          ; into the currently addressed directory entry
0FDF 1680                 mvi d,80h
                  copy$dir0:
0FE1 CDF80F               call copy$dir2
0FE4 0C                   inr c
                  copy$dir1:
0FE5 0DCA3E0C             dcr c! jz seek$copy
0FE9 7EA0C5               mov a,m! ana b! push b
0FEC 471AE67FB0           mov b,a! ldax d! ani 7fh! ora b! mov m,a
0FF2 C12313C3E5           pop b! inx h! inx d! jmp copy$dir1
                  copy$dir2:
0FF8 D5                   push d ; Save length for later
0FF9 0600                 mvi b,0 ; double index to bc
0FFB 2ADB1E               lhld info ; hl = source for data
0FFE 09                   dad b
0FFF 237ED624CC           inx h! mov a,m! sui '$'! cz set$submit$flag
1006 2BEB                 dcx h! xchg ; de=.fcb(c), source for copy
1008 CDFE0A               call getdptra ; hl=.buff(dptr), destination
100B C1                   pop b ; de=source, hl=dest, c=length
100C C9                   ret
```

```

                    set$submit$flag:
100D 11B31E1AF6             lxi d,ccp$flgs! ldax d! ori 1! stax d! ret

                    check$wild:
                            ; Check for ? in file name or type
1015 2ADB1E                 lhld info
                    check$wild0:   ; entry point used by rename
1018 CD2110C0               call chk$wild! rnz
101C 3E09C37F19             mvi a,9! jmp set$aret

                    chk$wild:
1021 0E0B                   mvi c,11
                    chk$wild1:
1023 233E3F96E6             inx h! mvi a,3fh! sub m! ani 7fh! rz
102A 0DC22310B7             dcr c! jnz chk$wild1! ora a! ret

                    copy$user$no:
1030 2ADB1E7E01             lhld info! mov a,m! lxi b,dskmap
1037 0977C9                 dad b! mov m,a! ret

                    rename:
                            ; Rename the file described by the first half of
                            ; the currently addressed file control block. The
                            ; new name is contained in the last half of the
                            ; currently addressed file control block.  The file
                            ; name and type are changed, but the reel number
                            ; is ignored.  The user number is identical.

                            ; Verify that the new file name does not exist.
                            ; Also verify that no wild chars exist in
                            ; either filename.

                    if MPM
                            call getatts! sta attributes
                    endif

                            ; Verify that no wild chars exist in 1st filename.
103A CD1510                 call check$wild

                    if BANKED
                            ; Check password of file to be renamed.
                            call chk$password! cnz chk$pw$error
                            ; Setup search to scan for xfcbs.
                            call init$xfcb$search
                    endif

                            ; Copy user number to 2nd filename
103D CD3010                 call copy$user$no
1040 22E31E                 shld searcha

                            ; Verify no wild chars exist in 2nd filename
1043 CD1810                 call check$wild0

                            ; Verify new filename does not already exist
1046 0E0C2AE31E             mvi c,extnum! lhld searcha! call searchi! call search1
```

```
1051 C27D                       jnz file$exists ; New filename exists

                        if BANKED
                                ; If an xfcb exists for the new filename, delete it.
                                call does$xfcb$exist! cnz delete11
                        endif

1054 CD3010                     call copy$user$no

                        if BANKED
                                call init$xfcb$search
                        endif

                                ; Search up to the extent field
1057 CD8B0E                     call search$extnum
105A C8                         rz
105B CDE60                      call check$rodir ; may be r/o file

                        if MPM
                                call chK$olist
                        endif

                                ; Copy position 0
                                rename0:
                                        ; not end of directory, rename next element
105E 0E101E0CCD                         mvi c,dsKmap! mvi e,extnum! call copy$dir
                                        ; element renamed, move to next
                                        .
1065 CDB620                             call fix$hash
1068 CD990E                             call searchn
106B C25E10                             jnz rename0
                                rename1:

                        if BANKED
                                        call does$xfcb$exist! jz copy$dir$loc
                                        call copy$user$no! jmp rename0
                        else
106E C3160E                             jmp copy$dir$loc
                        endif

                        indicators:
                                ; Set file indicators for current fcb
1071 CDD609                     call get$atts ; Clear f5' through f8'
1074 32D51C                     sta attributes

                        if BANKED
                                call chK$password! cnz chK$pw$error
                        endif

1077 CD8B0E                     call search$extnum ; through file type
107A C8                         rz

                        if MPM
                                call chK$olist
                        endif

```

```
                      indic0:
                              ; not end of directory, continue to change
107B 0E001E0C                 mvi c,0! mvi e,extnum ; Copy name
107F CDF80FCDF5               call copy$dir2! call move
1085 3AD51CE640               lda attributes! ani 40h! jz indic1

                              ; If interface att f6' set, dir fcb(s1) = fcb(cr)

108D E5CD0D0A7E               push h! call getfcba! mov a,m
1092 E12377                   pop h! inx h! mov m,a
                      indic1:
1095 CD3E0C                   call seek$copy
1098 CD990E                   call searchn
109B CA160E                   jz copy$dir$loc
109E C37B10                   jmp indic0

               open:
                      ; Search for the directory entry, copy to fcb
10A1 CD860E           call search$namlen
               open1:
10A4 C8               rz ; Return with lret=255 if end
                      ; not end of directory, copy fcb information
               open$copy:
10A5 CD200B5FE5       call setfwf! mov e,a! push h! dcx h! dcx h
10AC 56D5             mov d,m! push d ; Save extent# & module# with fcb write flag set
10AE CDFE0AEB         call getdptra! xchg ; hl = .buff(dptr)
10B2 2ADB1E           lhld info ; hl=.fcb(0)
10B5 0E20             mvi c,nxtrec ; length of move operation
10B7 CDF507           call move ; from .buff(dptr) to .fcb(0)
                      ; Note that entire fcb is copied, including indicators
10BA CD2D0E4F         call get$dir$ext! mov c,a
                      ; Restore module # and extent #
10BE D1E1732B2B       pop d! pop h! mov m,e! dcx h! dcx h! mov m,d
                      ; hl = .user extent#, c = dir extent#
                      ; above move set fcb(reccnt) to dir(reccnt)
                      ; if fcb ext < dir ext then fcb(reccnt) = fcb(reccnt) | 128
                      ; if fcb ext = dir ext then fcb(reccnt) = fcb(reccnt)
                      ; if fcb ext > dir ext then fcb(reccnt) = 0

               set$rc: ; hl=.fcb(ext), c=dirext
10C4 0600                     mvi b,0
10C6 EB21030019               xchg! lxi h,(reccnt-extnum)! dad d
                              ; Is fcb ext = dirext?
10CB 1A91CAD910               ldax d! sub c! jz set$rc2 ; yes
                              ; Is fcb ext > dirext?
10D0 78D2D710                 mov a,b! jnc set$rc1 ; yes - fcb(rc) = 0
                              ; fcb ext  < dirext
                              ; fcb(rc) = 128 | fcb(rc)
10D4 3E80B6                   mvi a,128! ora m
                      set$rc1:
10D7 77C9                     mov m,a! ret
                      set$rc2:
                              ; fcb ext = dirext
10D9 7EB7C0                   mov a,m! ora a! rnz ; ret if fcb(rc) ^= 0
                      set$rc3:
10DC 3600                     mvi m,0 ; required by function 99
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
10DE 3AC41CB7C8                    lda dminx! ora a! rz ; ret if no blks in fcb
10E3 3680C9                        mvi m,128! ret ; fcb(rc) = 128

                     mergezero:
                             ; hl = .fcb1(i), de = .fcb2(i),
                             ; if fcb1(i) = 0 then fcb1(i) := fcb2(i)
10E6 7E23B62BC0              mov a,m! inx h! ora m! dcx h! rnz ; return if = 0000
10EB 1A771323                ldax d! mov m,a! inx d! inx h ; low byte copied
10EF 1A771B2B                ldax d! mov m,a! dcx d! dcx h ; back to input form
10F3 C9                      ret

                     restore$rc:
                             ; hl = .fcb(extnum)
                             ; if fcb(rc) > 80h then fcb(rc) = fcb(rc) & 7fh
10F4 E5                      push h
10F5 11030019                lxi d,(reccnt-extnum)! dad d
10F9 7EFE81DA02              mov a,m! cpi 81h! jc restore$rc1
10FF E67F77                  ani 7fh! mov m,a
                     restore$rc1:
1102 E1C9                    pop h! ret

                     close:
                             ; Locate the directory element and re-write it
1104 AF327A07                xra a! sta lret

                     if MPM
                             sta dont$close
                     endif

1108 CDD80AC0                call nowrite! rnz ; Skip close if r/o disk
                             ; Check file write flag - 0 indicates written
110C CD090B                  call getmodnum ; fcb(modnum) in a
110F E680C0                  ani fwfmsk! rnz ; Return if bit remains set
                     close1:
1112 CD840CCAB9              call chk$inv$fcb! jz mergerr

                     if MPM
                             call set$fcb$cks$flag
                     endif

1118 CD2D0E4F                call get$dir$ext! mov c,a
111C 46C5                    mov b,m! push b
                             ; b = original extent, c = directory extent
                             ; Set fcb(ex) to directory extent
111E 71                      mov m,c
                             ; Recompute fcb(rc)
111F CDF410                  call restore$rc
                             ; Call set$rc if fcb ext > dir ext
1122 79B8DCC410              mov a,c! cmp b! cc set$rc
1127 CD3311                  call close$fcb
                             ; Restore original extent & reset fcb(rc)
112A CDFD09C1                call get$exta! pop b
112E 4E70C3C410              mov c,m! mov m,b! jmp set$rc ; Reset fcb(rc)

                     close$fcb:
                             ; Locate file
```

```
1133 CD860E             call search$namlen
1136 C8                 rz ; Return if not found
                        ; Merge the disk map at info with that at buff(dptr)
1137 011000CDAF         lxi b,dskmap! call get$fcb$adds
113D 0E10               mvi c,(fcblen-dskmap) ; length of single byte dm
                        merge0:
113F 3AC51CB7CA                 lda single! ora a! jz merged ; Skip to double
                                ; This is a single byte map
                                ; if fcb(i) = 0 then fcb(i) = buff(i)
                                ; if buff(i) = 0 then buff(i) = fcb(i)
                                ; if fcb(i) <> buff(i) then error
1146 7EB71AC24D                 mov a,m! ora a! ldax d! jnz fcbnzero
                                        ; fcb(i) = 0
114C 77                                 mov m,a ; fcb(i) = buff(i)
                                fcbnzero:
114D B7C25311                   ora a! jnz buffnzero
                                        ; buff(i) = 0
1151 7E12                               mov a,m! stax d ; buff(i)=fcb(i)
                                buffnzero:
1153 BEC2B911                   cmp m! jnz mergerr ; fcb(i) = buff(i)?
1157 C36F11                     jmp dmset ; if merge ok
                        merged:
                                ; This is a double byte merge operation
115A CDE610                     call mergezero ; buff = fcb if buff 0000
115D EBCDE610EB                 xchg! call mergezero! xchg ; fcb = buff if fcb 0000
                                ; They should be identical at this point
1162 1ABEC2B911                 ldax d! cmp m! jnz mergerr ; low same?
1167 1323                       inx d! inx h ; to high byte
1169 1ABEC2B911                 ldax d! cmp m! jnz mergerr ; high same?
                                ; merge operation ok for this pair
116E 0D                         dcr c ; extra count for double byte
                        dmset:
116F 1323                       inx d! inx h ; to next byte position
1171 0DC23F11                   dcr c! jnz merge0 ; for more
                                ; end of disk map merge, check record count
                                ; de = .buff(dptr)+32, hl = .fcb(32)

1175 EB01ECFF09                 xchg! lxi b,-(fcblen-extnum)! dad b! push h
117B CD2D0ED1                   call get$dir$ext! pop d

                                ; hl = .fcb(extnum), de = .buff(dptr+extnum)

117F CDBF11                     call compare$extents

                                ; b=1 -> fcb(ext) ^= dir ext = buff(ext)
                                ; b=2 -> fcb(ext) = dir ext ^= buff(ext)
                                ; b=3 -> fcb(ext) = dir ext = buff(ext)

                                ; fcb(ext), buff(ext) = dir ext
1182 7712C5                     mov m,a! stax d! push b

1185 01030009EB                 lxi b,(reccnt-extnum)! dad b! xchg! dad b
118B C1                         pop b

                                ; hl = .buff(rc) , de = .fcb(rc)
```

```
118C 05CAA011                               dcr b! jz mrg$rc1 ; fcb(rc) = buff(rc)

1190 05CAA111                               dcr b! jz mrg$rc2 ; buff(rc) = fcb(rc)

1194 1ABEDAA011                             ldax d! cmp m! jc mrg$rc1 ; Take larger rc
1199 B7C2A111                               ora a! jnz mrg$rc2
119D CDDC10                                 call set$rc3

11A0 EB                         mrg$rc1: xchg

11A1 1A77                       mrg$rc2: ldax d! mov m,a

                      if MPM
                                            lda dont$close! ora a! rnz
                      endif

                                            ; Set t3' off indicating file update
11A3 CDFE0A110B                             call getdptra! lxi d,11! dad d
11AA 7EE67F77                               mov a,m! ani 7fh! mov m,a
11AE CD200B                                 call setfwf
11B1 0E01CD320D                             mvi c,1! call scandm$b ; Set 2nd ALV vector
11B6 C33E0C                                 jmp seek$copy ; OK to 'wrdir' here - 1.4 compat
                                            ; ret
                                mergerr:
                                            ; elements did not merge correctly
11B9 CD790C                                 call make$fcb$inv
11BC C3260F                                 jmp lret$eq$ff

                      compare$extents:
11BF 0601BEC0                   mvi b,1! cmp m! rnz
11C3 04EBBEEBC0                 inr b! xchg! cmp m! xchg! rnz
11C8 04C9                       inr b! ret

                      set$xdcnt:
11CA 21FFFF22DF                 lxi h,0ffffh! shld xdcnt! ret

                      set$dcnt$dblk:
11D1 2ADF1C                     lhld xdcnt
                      set$dcnt$dblk1:
11D4 3EFCA5                     mvi a,1111$1100b! ana l
11D7 6F2B22E11E                 mov l,a! dcx h! shld dcnt

                      if MPM
                                lhld xdblk! shld dblk
                      endif

11DC C9                         ret

                      if MPM

                      sdcnt$eq$xdcnt:
                                lxi h,sdcnt! lxi d,xdcnt! mvi c,4
                                jmp move
                      endif

                      make:
```

```
                                ; Create a new file by creating a directory entry
                                ; then opening the file

11DD 21DF1CCD97                 lxi h,xdcnt! call test$ffff! cnz set$dcnt$dblk

11E6 2ADB1EE5                   lhld info! push h ; Save fcb address, Look for E5
11EA 21851C22DB                 lxi h,efcb! shld info ; info = .empty
11F0 0E01                       mvi c,1

11F2 CD730ECD99                 call searchi! call searchn

                                ; zero flag set if no space
11F8 E1                         pop h ; Recall info address
11F9 22DB1E                     shld info ; in case we return here
11FC C8                         rz ; Return with error condition 255 if not found

                      if BANKED
                                ; Return early if making an xfcb
                                lda make$xfcb! ora a! rnz
                      endif

                                ; Clear the remainder of the fcb
                                ; Clear s1 byte
11FD 110D001972                 lxi d,13! dad d! mov m,d! inx h
                                ; Clear and save file write flag of modnum
1203 7EF5E5E63F                 mov a,m! push a! push h! ani 3fh! mov m,a! inx h
120A 3E01                       mvi a,1
120C 0E11                       mvi c,fcblen-namlen ; number of bytes to fill
                                make0:
120E 72230DC20E                         mov m,d! inx h! dcr c! jnz make0
1214 3D4ACCA717                         dcr a! mov c,d! cz get$dtba
1219 B70E0ACA0E                         ora a! mvi c,10! jz make0
121F CD340B                     call setcdr ; may have extended the directory
                                ; Now copy entry to the directory
1222 0E00112000                 mvi c,0! lxi d,fcblen! call copy$dir0
                                ; and restore the file write flag
122A E1F177                     pop h! pop a! mov m,a
                                ; and set the fcb write flag to '1'
122D CDB620                     call fix$hash
1230 C3200B                     jmp setfwf

                      open$reel:
                                ; Close the current extent, and open the next one
                                ; if possible.  rmf is true if in read mode

                      if BANKED
                                call reset$copy$cr$only
                      endif

1233 CDFD09                     call getexta
1236 7E4F                       mov a,m! mov c,a
1238 0CCD1C0E                   inr c! call compext
123C CA9912                     jz open$reel3
123F E5C5                       push h! push b
1241 CD0411                     call close
1244 C1E1                       pop b! pop h
```

```
1246 3A7A073CC8         lda lret! inr a! rz
124B 3E1FA177           mvi a,maxext! ana c! mov m,a ; Incr extent field
                        ; Advance to module & save
124F 23237E32C9         inx h! inx h! mov a,m! sta save$mod
1255 C25F12             jnz open$reel0 ; Jump if in same module

                        open$mod:
                                ; Extent number overflow, go to next module
1258 34                         inr m ; fcb(modnum)=++1
                                ; Module number incremented, check for overflow

1259 7EE63F                     mov a,m! ani 3fh ; Mask high order bits

125C CA8812                     jz open$r$err ; cannot overflow to zero

                                ; otherwise, ok to continue with new module
                        open$reel0:
125F CDCA11                     call set$xdcnt ; Reset xdcnt for make

                if MPM
                                call set$sdcnt
                endif

1262 CD860E                     call search$namlen ; Next extent found?
1265 C27812                     jnz open$reel1
                                        ; end of file encountered
1268 3AC11C3C                           lda rmf! inr a ; 0ffh becomes 00 if read
126C CA8812                             jz open$r$err ; sets lret = 1
                                        ; Try to extend the current file
126F CDDD11                             call make
                                        ; cannot be end of directory
1272 CA8812                             jz open$r$err ; with lret = 1

                if MPM
                                        call fix$olist$item
                                        call set$fcb$cks$flag
                endif

1275 C37B12                             jmp open$reel2
                                open$reel1:
                                        ; not end of file, open
1278 CDA510                             call open$copy

                if MPM
                                        call set$fcb$cks$flag
                endif

                                open$reel2:

                if not MPM
127B CDFF0B                             call set$lsn
                endif

127E CD160A                             call getfcb ; Set parameters
1281 AF32CA1CC3                         xra a! sta vrecord! jmp sta$ret ; lret = 0
                                        ; ret ; with lret = 0
                        open$r$err:
                                ; Restore module and extent
1288 CD090B3AC9                 call getmodnum! lda save$mod! mov m,a
128F 2B2B7E3DE6                 dcx h! dcx h! mov a,m! dcr a! ani 1fh
1295 77C3CF06                   mov m,a! jmp setlret1 ; lret = 1

                        open$reel3:
1299 34                         inr m ; fcb(ex) = fcb(ex) + 1
129A CD2D0E4F                   call get$dir$ext! mov c,a
                                ; Is new extent beyond dir$ext?
129E BED2AB12                   cmp m! jnc open$reel4 ; no
12A2 35                         dcr m ; fcb(ex) = fcb(ex) - 1
                                ; Is this a read fx?
12A3 3AC11C3CCA                 lda rmf! inr a! jz set$lret1 ; yes - Don't advance ext
12AA 34                         inr m ; fcb(ex) = fcb(ex) + 1
                        open$reel4:
12AB CDF410                     call restore$rc
12AE CDC410C37B                 call set$rc! jmp open$reel2

                seqdiskread:
                diskread:       ; (may enter from seqdiskread)
12B4 CD8A0C             call tst$inv$fcb ; Check for valid fcb
12B7 3EFF32C11C         mvi a,true! sta rmf ; read mode flag = true (open$reel)

                if MPM
                        sta dont$close
                endif

                        ; Read the next record from the current fcb
12BC CD160A             call getfcb ; sets parameters for the read
                diskread0:
12BF 3ACA1C21C7         lda vrecord! lxi h,rcount! cmp m ; vrecord-rcount
                        ; Skip if rcount > vrecord
12C6 DAD812             jc recordok

                if MPM
                                call test$disk$fcb! jnz diskread0
                                lda vrecord
                endif

                                ; not enough records in the extent
                                ; record count must be 128 to continue
12C9 FE80                       cpi 128 ; vrecord = 128?
12CB C2CF06                     jnz setlret1 ; Skip if vrecord<>128
12CE CD3312                     call open$reel ; Go to next extent if so
                                ; Check for open ok
12D1 3A7A07B7C2                 lda lret! ora a! jnz setlret1 ; Stop at eof
                        recordok:
                                ; Arrive with fcb addressing a record to read

                if BANKED
                                call set$copy$cr$only
                endif

12D8 CD9E09                     call index ; Z flag set if arecord = 0

```

```
               if MPM
                              jnz recordok1
                              call test$disk$fcb! jnz diskread0
               endif

12DB CACF06                   jz setlret1 ; Reading unwritten data
                      recordok1:
                              ; Record has been allocated, read it
12DE CDB009                   call atran ; arecord now a disk address
12E1 CD0413                   call check$nprs
12E4 DA400A                   jc setfcb
12E7 C2F612                   jnz read$deblock

12EA CD520C                   call setdata
12ED CDCA08                   call seek ; to proper track,sector

               if BANKED
                              mvi a,1! call setbnkf
               endif

12F0 CD5F08                   call rdbuff ; to dma address
12F3 C3400A                   jmp setfcb ; Replace parameter

               read$deblock:
12F6 21000022BB       lxi h,0! shld last$block
12FC 3E01CD021D       mvi a,1! call deblock$dta
1301 C3400A           jmp setfcb

               check$nprs:
                      ;
                      ; on exit,  c flg          -> no i/o operation
                      ;            z flg & ~c flg -> direct(physical) i/o operation
                      ;           ~z flg & ~c flg -> indirect(deblock) i/o operation
                      ;
                      ;          Dir$cnt contains the number of 128 byte records
                      ;          to transfer directly.  This routine sets dir$cnt
                      ;          when initiating a sequence of direct physical
                      ;          i/o operations.  Dir$cnt is decremented each
                      ;          time check$nprs is called during such a sequence.
                      ;
                      ; Is direct transfer operation in progress?
1304 3AB81C47         lda blk$off! mov b,a
1308 3AB51C4FA0       lda phy$msk! mov c,a! ana b! push a
130E 3ABD1CFE02       lda dir$cnt! cpi 2! jc check$npr1 ; no
                      ; yes - Decrement direct record count
1316 3D32BD1C         dcr a! sta dir$cnt
                      ; Are we at a new physical record?
131A F137C0           pop a! stc! rnz ; no - ret with c flg set
                      ; Perform physical i/o operation
131D AFC9             xra a! ret ; Return with z flag set and c flag reset
               check$npr1:
                      ; Are we in mid-physical record?
131F F1CA2913         pop a! jz check$npr11 ; no
               check$npr1a:
                      ; Is phymsk = 0?
1323 79B7C8           mov a,c! ora a! rz ; yes - Don't deblock
```

```
                          check$npr1b:
                                    ; Deblocking required
1326 F601C9                         ori 1! ret ; ret with z flg reset and c flg reset
                          check$npr11:
1329 792F57                         mov a,c! cma! mov d,a ; d = ~phy$msk
132C 21CA1C                         lxi h,vrecord
                                    ; Is mult$num < 2?
132F 3ABE1CFE02                     lda mult$num! cpi 2! jc check$npr1a ; yes
1337 86FE80DA3F                     add m! cpi 80h! jc check$npr2
133D 3E80                           mvi a,80h
                          check$npr2: ; a = min(vrecord + mult$num),80h) = x
133F C5                             push b ; Save low(arecord) & blkmsk, phymsk
1340 46367F                         mov b,m! mvi m,7fh ; vrecord = 7f
1343 C5                             push b ; Save vrecord
1344 E5                             push h ; Save .vrecord
1345 F5                             push a ; Save x
1346 3AA81C5F1C                     lda blkmsk! mov e,a! inr e! cma! ana b! mov b,a
                                    ; b = vrecord & ~blkmsk
                                    ; e = blkmsk + 1
134E E1                             pop h ; h = x
                                    ; Is this a read function?
134F 3AC11CB7CA                     lda rmf! ora a! jz check$npr21 ; no
                                    ; Is rcount & ~phymsk < x?
1356 3AC71CA2BC                     lda rcount! ana d! cmp h! jc check$npr23 ; yes
                          check$npr21:
135E 7C                             mov a,h ; a = x
                          check$npr23:
135F 90                             sub b ; a = a - vrecord & ~blkmsk
1360 4F                             mov c,a ; c = max # of records from beginning of curr blk
                                    ; Is c < blkmsk+1?
1361 BBDA6613                       cmp e! jc check$npr8 ; yes

                          if BANKED
                                    push b ; c = max # of records
                                    ; Compute maximum disk map position
                                    call dm$position
                                    mov b,a ; b = index of last block in extent
                                    ; Does the last block # = the current block #?
                                    lda dminx! cmp b! mov e,a! jz check$npr5 ; yes
                                    ; Compute # of blocks in sequence
                                    mov c,a! push b! mvi b,0
                                    call get$dm ; hl = current block #
                          check$npr4:
                                    ; Get next block #
                                    push h! inx b! call get$dm
                                    pop d! inx d
                                    ; Does next block # = previous block # + 1?
                                    mov a,d! sub h! mov d,a
                                    mov a,e! sub l! ora d! jz check$npr4 ; yes
                                    ; Is next block # = 0?
                                    mov a,h! ora l! jnz check$npr45 ; no
                                    ; Is this a read function?
                                    lda rmf! ora a! jnz check$npr45 ; no
                                    ; Is next block # > maxall?
                                    lhld maxall! mov a,l! sub e
                                    mov a,h! sbb d! jc check$npr45 ; yes
```

```
                                ; Is next block # allocated?
                                push b! push d! mov b,d! mov c,e
                                call getallocbit! pop h! pop b
                                rar! jnc check$npr4 ; no - it will be later
                        check$npr45:
                                dcr c! pop d
                                ; Is max dm position less than c?
                                mov a,d! cmp c! jc check$npr5 ; yes
                                mov a,c ; no
                        check$npr5: ; a = index of last block
                                sub e! mov b,a! inr b ; b = # of consecutive blks
                                lda blkmsk! inr a! mov c,a
                        check$npr6:
                                dcr b! jz check$npr7
                                add c! jmp check$npr6
                        check$npr7:
                                pop b
                                mov b,c ; b = max # of records
                                mov c,a ; c = (# of consecutive blks)*(blkmsk+1)
                                lda rmf! ora a! jz check$npr8
                                mov a,b! cmp c! jc check$npr9
                        else
1365 4B                         mov c,e ; multis-sector max = 1 block in non-banked systems
                        endif

                        check$npr8:
1366 79                         mov a,c
                        check$npr9:
                                ; Restore vrecord
1367 E1C170                     pop h! pop b! mov m,b
136A C1                         pop b
                                ; a = max # of consecutive records including current blk
                                ; b = low(arecord) & blkmsk
                                ; c = phymsk
                                ; Is mult$num > a - b
136B 21BE1C56                   lxi h,mult$num! mov d,m
136F 90BAD27513                 sub b! cmp d! jnc check$npr10
1374 57                         mov d,a ; yes - use smaller value to compute dir$cnt
                        check$npr10:
                                ; Does this operation involve at least 1 physical record?
1375 792FA232BD                 mov a,c! cma! ana d! sta dir$cnt! jz check$npr1b ; Deblocking required
                                ; Flush any pending buffers before doing multiple reads
137E F53AC11CB7                 push a! lda rmf! ora a! jz check$npr10a
1386 CDBA1ACD52                 call flushx! call setdata
                        check$npr10a:
138C F167                       pop a! mov h,a ; Save # of 128 byte records
                                ; Does this operation involve more than 1 physical record?
                                ; Register h contains number of 128 byte records
138E CD5B097C                   call shr$physhf! mov a,h
1392 FE014FC445                 cpi 1! mov c,a! cnz mult$iof ; yes - Make bios call
1398 AFC9                       xra a! ret ; Return with z flg set

                        if MPM

                        test$unlocked:
                                lda high$ext! ani 80h! ret
```

```

                      test$disk$fcb:
                              call test$unlocked! rz
                              lda dont$close! ora a! rz
                              call close1
                      test$disk$fcb1:
                              pop d
                              lxi h,lret! inr m! mvi a,11! jz sta$ret
                              mvi m,0
                              push d
                              call getrcnta! mov a,m! sta rcount ; Reset rcount
                              xra a! sta dont$close
                              inr a! ret
                      endif

                      reset$fwf:
139A CD090B                   call getmodnum ; hl=.fcb(modnum), a=fcb(modnum)
                              ; Reset the file write flag to mark as written fcb
139D E67F                     ani (not fwfmsk) and 0ffh ; bit reset
139F 77                       mov m,a ; fcb(modnum) = fcb(modnum) and 7fh
13A0 C9                       ret

                      set$filewf:
13A1 CD090BE640               call getmodnum! ani 0100$0000b! push a
13A7 7EF64077F1               mov a,m! ori 0100$0000b! mov m,a! pop a! ret

                      seqdiskwrite:
                      diskwrite:      ; (may enter here from seqdiskwrite above)
13AD 3E0032C11C               mvi a,false! sta rmf ; read mode flag
                              ; Write record to currently selected file

13B2 CDF70A                   call check$write ; in case write protected

                      if BANKED
                              lda xfcb$read$only! ora a
                              mvi a,3! jnz set$aret
                      endif

13B5 3AD71C                   lda high$ext

                      if MPM
                              ani 0100$0000b
                      else
13B8 B7                       ora a
                      endif

                              ; Z flag reset if r/o mode
13B9 3E03C27F19               mvi a,3! jnz set$aret

13BE 2ADB1E                   lhld info ; hl = .fcb(0)
13C1 CDE90A                   call check$rofile ; may be a read-only file

13C4 CD8A0C                   call tst$inv$fcb ; Test for invalid fcb

13C7 CDE817                   call update$stamp

```

```
13CA CD160A             call getfcb ; to set local parameters
13CD 3ACA1CFE80         lda vrecord! cpi lstrec+1 ; vrecord-128
13D2 DADD13             jc diskwrite0
13D5 CD3312             call open$reel ; vrecord = 128, try to open next extent
13D8 3A7A07B7C0         lda lret! ora a! rnz ; no available fcb
                disk$write0:

                if MPM
                        mvi a,0ffh! sta dont$close
                disk$write1:

                endif

                        ; Can write the next record, so continue
13DD CD9E09             call index ; Z flag set if arecord = 0
13E0 CAFB13             jz diskwrite2
                        ; Was the last write operation for the same block & drive?
13E3 21CC1C11BA         lxi h,adrive! lxi d,last$drive! mvi c,3
13EB CDEB07CAF6         call compare! jz diskwrite15 ; yes
                        ; no - force preread in blocking/deblocking
13F1 3EFF32B91C         mvi a,0ffh! sta last$off
                diskwrite15:

                if MPM
                        ; If file is unlocked, verify record is not locked
                        ; Record has to be allocated to be locked
                        call test$unlocked! jz not$unlocked
                        call atran! mov c,a
                        lda mult$cnt! mov b,a! push b
                        call test$lock! pop b
                        xra a! mov c,a! push b
                        jmp diskwr10
                not$unlocked:
                        inr a
                endif

13F6 0E00               mvi c,0 ; Marked as normal write operation for wrbuff
13F8 C34F14             jmp diskwr1
                diskwrite2:

                if MPM
                                call test$disk$fcb! jnz diskwrite1
                endif

                if BANKED
                                call reset$copy$cr$only
                endif

                                ; not allocated
                                ; The argument to getblock is the starting
                                ; position for the disk search, and should be
                                ; the last allocated block for this file, or
                                ; the value 0 if no space has been allocated
13FB CD6209                     call dm$position
13FE 32C41C                     sta dminx ; Save for later
1401 010000                     lxi b,0000h ; May use block zero
```

```
1404 B7CA0F14                   ora a! jz nopblock ; Skip if no previous block
                                        ; Previous block exists at a
1408 4F0B                               mov c,a! dcx b ; Previous block # in bc
140A CD8A09                             call getdm ; Previous block # to hl
140D 444D                               mov b,h! mov c,l ; bc=prev block#
                                nopblock:
                                        ; bc = 0000, or previous block #
140F CDA00F                             call get$block ; block # to hl
                                ; Arrive here with block# or zero
1412 7DB4C21C14                 mov a,l! ora h! jnz blockok
                                        ; Cannot find a block to allocate
1417 3E02C3CB06                         mvi a,2! jmp sta$ret ; lret=2
                                blockok:

                if MPM
                                call set$fcb$cks$flag
                endif

                                ; allocated block number is in hl
141C 22CD1C22BB                 shld arecord! shld last$block! xra a! sta last$off
1426 3ACC1C32BA                 lda adrive! sta lastdrive
142C EB                         xchg ; block number to de
142D 2ADB1E0110                 lhld info! lxi b,dskmap! dad b ; hl=.fcb(dskmap)
1434 3AC51CB7                   lda single! ora a ; Set flags for single byte dm
1438 3AC41C                     lda dminx ; Recall dm index
143B CA4514                     jz allocwd ; Skip if allocating word
                                        ; Allocating a byte value
143E CD040B73                           call addh! mov m,e ; single byte alloc
1442 C34D14                             jmp diskwru ; to continue
                                allocwd:
                                ; Allocate a word value
1445 4F0600                             mov c,a! mvi b,0 ; double(dminx)
1448 0909                               dad b! dad b ; hl=.fcb(dminx*2)
144A 732372                             mov m,e! inx h! mov m,d ; double wd
                                diskwru:
                                ; disk write to previously unallocated block
144D 0E02                       mvi c,2 ; marked as unallocated write
                        diskwr1:
                        ; Continue the write operation of no allocation error
                        ; c = 0 if normal write, 2 if to prev unalloc block
144F C5                 push b ; Save write flag
1450 CDB009             call atran ; arecord set
                diskwr10:
1453 3ADF1EFE28                 lda fx! cpi 40! jnz diskwr11 ; fx ~= wrt rndm zero fill
145B 793D3DC2B1                 mov a,c! dcr a! dcr a! jnz diskwr11 ; old allocation

                                ; write random zero fill + new block

1461 C1F5                       pop b! push a ; zero write flag
1463 2ACD1CE5                   lhld arecord! push h
1467 21B51C5E1C                 lxi h,phymsk! mov e,m! inr e! mov d,a! push d
146E 2A9E1C                     lhld dirbcba

                if BANKED
                                mov e,m! inx h! mov d,m! xchg
                fill00:
```

```
                              push h! call get$next$bcba! pop d! jnz fill00
                              xchg
               endif

                              ; Force prereads in blocking/deblocking
                              ; Discard BCB
1471 3D32B91C77               dcr a! sta last$off! mov m,a
1476 CD580C                   call setdir1 ; Set dma to BCB buffer
                              ; Zero out BCB buffer
1479 D1D5AF                   pop d! push d! xra a
                      fill0:
147C 772314F27C               mov m,a! inx h! inr d! jp fill0
1482 571DC27C14               mov d,a! dcr e! jnz fill0
                              ; Write 1st physical record of block
1487 2AD31C0E02               lhld arecord1! mvi c,2
                      fill1:
148C 22CD1CC5CD               shld arecord! push b! call discard$data$bcb
1493 CDCA08                   call seek

               if BANKED
                              xra a! call setbnkf
               endif

1496 C1CD6A08                 pop b! call wrbuff
149A 2ACD1CD1D5               lhld arecord! pop d! push d
                              ; Continue writing until blkmsk & arecord = 0
149F 193AA81CA5               dad d! lda blkmsk! ana l! mvi c,0! jnz fill1
                              ; Restore arecord
14A9 E1E122CD1C               pop h! pop h! shld arecord

14AE CD520C                   call setdata ; Restore dma
                      diskwr11:

14B1 D13ACA1C57       pop d! lda vrecord! mov d,a ; Load and save vrecord
14B6 D5CD0413         push d! call check$nprs

14BA DAE214           jc dont$write
14BD CAC814           jz write

14C0 3E02             mvi a,2 ; deblock write code
14C2 CD021D           call deblock$dta
14C5 C3E214           jmp dont$write
               write:
14C8 CD520C           call setdata
14CB CDCA08           call seek

               if BANKED
                      mvi a,1! call setbnkf
               endif

                      ; Discard matching BCB if write is direct
14CE CD080C           call discard$data$bcb

                      ; Set write flag to zero if arecord & blkmsk ^= 0

14D1 C1C53ACD1C       pop b! push b! lda arecord
```

```
14D6 21A81CA6CA                 lxi h,blkmsk! ana m! jz write0
14DD 0E00                       mvi c,0
                        write0:
14DF CD6A08                     call wrbuff

                        dont$write:
14E2 C1                         pop b ; c = 2 if a new block was allocated, 0 if not
                                ; Increment record count if rcount<=vrecord
14E3 7821C71CBE                 mov a,b! lxi h,rcount! cmp m ; vrecord-rcount
14E8 DAEF14                     jc diskwr2
                                        ; rcount <= vrecord
14EB 7734                               mov m,a! inr m ; rcount = vrecord+1

                        if MPM
                                        call test$unlocked! jz write1

                                        ; for unlocked files
                                        ;   rcount = rcount & (~ blkmsk) + blkmsk + 1

                                        lda blkmsk! mov b,a! inr b! cma! mov c,a
                                        mov a,m! dcr a! ana c! add b! mov m,a
                                write1:
                        endif

14ED 0E02                               mvi c,2 ; Mark as record count incremented
                                diskwr2:
                                ; a has vrecord, c=2 if new block or new record#
14EF 0D0DC2F714                 dcr c! dcr c! jnz noupdate
14F4 CD9A13                             call reset$fwf

                        if MPM
                                        call test$unlocked! jz noupdate
                                        lda rcount! call getrcnta! mov m,a
                                        call close
                                        call test$disk$fcb1
                        endif

                        noupdate:
                                ; Set file write flag if reset
14F7 CDA113                     call set$filewf

                        if BANKED
                                jnz disk$write3
                                ; Reset fcb file write flag to ensure t3' gets
                                ; reset by the close function
                                call reset$fwf
                                call reset$copy$cr$only
                                jmp setfcb
                        disk$write3:
                                call set$copy$cr$only
                        else
14FA CC9A13                     cz reset$fwf
                        endif
14FD C3400A                     jmp setfcb ; Replace parameters
                                ; ret
```

```
                     rseek:
                        ; Random access seek operation, c=0ffh if read mode
                        ; fcb is assumed to address an active file control block
                        ; (1st block of FCB = 0ffffh if previous bad seek)
1500 C5                 push b ; Save r/w flag
1501 2ADB1EEB           lhld info! xchg ; de will hold base of fcb
1505 21210019                   lxi h,ranrec! dad d ; hl=.fcb(ranrec)
1509 7EE67FF5                   mov a,m! ani 7fh! push psw ; record number
150D 7E17                       mov a,m! ral ; cy=lsb of extent#
150F 237E17E61F                 inx h! mov a,m! ral! ani 11111b ; a=ext#
1514 4F                         mov c,a ; c holds extent number, record stacked

1515 7EE6F023B6                 mov a,m! ani 1111$0000b! inx h! ora m
151A 0F0F0F0F47                 rrc! rrc! rrc! rrc! mov b,a
                                ; b holds module #

                                ; Check high byte of ran rec <= 3
151F 7E                         mov a,m
1520 E6FCE12E06                 ani 1111$1100b! pop h! mvi l,6! mov a,h

                                ; Produce error 6, seek past physical eod
1526 C2C915                     jnz seekerr

                                ; otherwise, high byte = 0, a = sought record
1529 21200019                   lxi h,nxtrec! dad d ; hl = .fcb(nxtrec)
152D 77                         mov m,a ; sought rec# stored away

                        ; Arrive here with b=mod#, c=ext#, de=.fcb, rec stored
                        ; the r/w flag is still stacked.  compare fcb values

152E 3ADF1EFE63                 lda fx! cpi 99! jz rseek3
                                ; Check module # first
1536 D5CD840CD1                 push d! call chk$inv$fcb! pop d! jz ranclose
153E 210E001978                 lxi h,modnum! dad d! mov a,b ; b=seek mod#
1543 96E63FC277                 sub m! ani 3fh! jnz ranclose ; same?
                                ; Module matches, check extent
1549 210C0019                   lxi h,extnum! dad d
154D 7EB9CABD15                 mov a,m! cmp c! jz seekok2 ; extents equal
1552 CD1C0EC277                 call compext! jnz ranclose
                                ; Extent is in same directory fcb
1558 C5CD2D0EC1                 push b! call get$dir$ext! pop b
155D B9D26C15                   cmp c! jnc rseek2 ; jmp if dir$ext > ext
1561 D1D51CC26C                 pop d! push d! inr e! jnz rseek2 ; jmp if write fx
1567 1CD1C3CF06                 inr e! pop d! jmp set$lret1 ; error - reading unwritten data
                        rseek2:
156C 71                         mov m,c ; fcb(ext) = c
156D 4F                         mov c,a ; c = dir$ext
                                ; hl=.fcb(ext),c=dir ext
156E CDF410                     call restore$rc
1571 CDC410                     call set$rc
1574 C3BD15                     jmp seekok1
                        ranclose:
1577 C5D5                       push b! push d ; Save seek mod#,ext#, .fcb
1579 CD0411                     call close ; Current extent closed
157C D1C1                       pop d! pop b ; Recall parameters and fill
157E 2E03                       mvi l,3 ; Cannot close error #3
```

```
1580 3A7A073CCA               lda lret! inr a! jz seekerr
                       rseek3:
1587 CDCA11                   call set$xdcnt ; Reset xdcnt for make

               if MPM
                              call set$sdcnt
               endif

158A 210C0019E5               lxi h,extnum! dad d! push h
158F 5671                     mov d,m! mov m,c ; fcb(extnum)=ext#
1591 23237E5FD5               inx h! inx h! mov a,m! mov e,a! push d
1596 E640B077                 ani 040h! ora b! mov m,a
                              ; fcb(modnum)=mod#
159A CDA110                   call open ; Is the file present?
159D 3A7A073CC2               lda lret! inr a! jnz seekok ; Open successful?
                              ; Cannot open the file, read mode?
15A4 D1E1C1                   pop d! pop h! pop b ; r/w flag to c (=0ffh if read)
15A7 C5E5D5                   push b! push h! push d ; Restore stack
15AA 2E04                     mvi l,4 ; Seek to unwritten extent #4
15AC 0C                       inr c ; becomes 00 if read operation
15AD CAC215                   jz badseek ; Skip to error if read operation
                              ; Write operation, make new extent
15B0 CDDD11                   call make
15B3 2E05                     mvi l,5 ; cannot create new extent #5
15B5 CAC215                   jz badseek ; no dir space

               if MPM
                              call fix$olist$item
               endif

                              ; file make operation successful
                       seekok:
15B8 C1C1                     pop b! pop b ; Discard top 2 stacked items

               if MPM
                              call set$fcb$cks$flag
               else
15BA CDFF0B                   call set$lsn
               endif

                       seekok1:

               if BANKED
                              call reset$copy$cr$only
               endif

                       seekok2:
15BD C1                       pop b ; Discard r/w flag or .fcb(ext)
15BE AFC3CB06                 xra a! jmp sta$ret ; with zero set
                       badseek:
                              ; Restore fcb(ext) & fcb(mod)
15C2 D1E3                     pop d! xthl ; Save error flag
15C4 72232373                 mov m,d! inx h! inx h! mov m,e
15C8 E1                       pop h ; Restore error flag
                       seekerr:
                        if BANKED
                                        call reset$copy$cr$only ; Z flag set
                                        inr a ; Reset Z flag
                        endif

15C9 C1                                 pop b ; Discard r/w flag
15CA 7DC3CB06                           mov a,l! jmp sta$ret ; lret=#, nonzero

                        randiskread:
                                ; Random disk read operation
15CE 0EFF                       mvi c,true ; marked as read operation
15D0 CD0015                     call rseek
15D3 CCB412                     cz diskread ; if seek successful
15D6 C9                         ret

                        randiskwrite:
                                ; Random disk write operation
15D7 0E00                       mvi c,false ; marked as write operation
15D9 CD0015                     call rseek
15DC CCAD13                     cz diskwrite ; if seek successful
15DF C9                         ret

                        compute$rr:
                                ; Compute random record position for getfilesize/setrandom
15E0 EB19                       xchg! dad d
                                ; de=.buf(dptr) or .fcb(0), hl = .f(nxtrec/reccnt)
15E2 4E0600                     mov c,m! mvi b,0 ; bc = 0000 0000 ?rrr rrrr
15E5 210C00197E                 lxi h,extnum! dad d! mov a,m! rrc! ani 80h ; a=e000 0000
15ED 814F3E0088                 add c! mov c,a! mvi a,0! adc b! mov b,a
                                ; bc = 0000 000? errrr rrrr
15F3 7E0FE60F80                 mov a,m! rrc! ani 0fh! add b! mov b,a
                                ; bc = 000? eeee errrr rrrr
15F9 210E00197E                 lxi h,modnum! dad d! mov a,m ; a=xxmm mmmm
15FE 87878787                   add a! add a! add a! add a ; cy=m a=mmmm 0000

1602 B78047F5                   ora a! add b! mov b,a! push psw ; Save carry
1606 7E1F1F1F1F                 mov a,m! rar! rar! rar! rar! ani 0000$0011b ; a=0000 00mm
160D 6FF13E008D                 mov l,a! pop psw! mvi a,0! adc l ; Add carry
1612 C9                         ret

                        compare$rr:
1613 5F                         mov e,a ; Save cy
1614 79965723                   mov a,c! sub m! mov d,a! inx h ; lst byte
1618 789E23                     mov a,b! sbb m! inx h ; middle byte
161B F5B257F1                   push a! ora d! mov d,a! pop a
161F 7B9E                       mov a,e! sbb m ; carry if .fcb(ranrec) > directory
1621 C9                         ret

                        set$rr:
1622 732B702B71                 mov m,e! dcx h! mov m,b! dcx h! mov m,c! ret

                        getfilesize:
                                ; Compute logical file size for current fcb
                                ; Zero the receiving ranrec field
1628 CDF509E5                   call get$rra! push h ; Save position
162C 7223722372                 mov m,d! inx h! mov m,d! inx h! mov m,d ; =00 00 00
```

```
1631 CD8B0E             call search$extnum
                        getsize:
1634 CA5316                     jz setsize
                                ; current fcb addressed by dptr
1637 CDFE0A110F                 call getdptra! lxi d,reccnt ; ready for compute size
163D CDE015                     call compute$rr
                                ; a=0000 00mm bc = mmmm eeee errr rrrr
                                ; Compare with memory, larger?
1640 E1E5                       pop h! push h ; Recall, replace .fcb(ranrec)
1642 CD1316D422                 call compare$rr! cnc set$rr
1648 CD990E                     call searchn
164B 3E00327A07                 mvi a,0! sta aret
1650 C33416                     jmp getsize
                        setsize:

1653 E1                 pop h ; Discard .fcb(ranrec)
1654 C9                 ret

                setrandom:
                        ; Set random record from the current file control block
1655 EB112000           xchg! lxi d,nxtrec ; Ready params for computesize
1659 CDE015             call compute$rr ; de=info, a=0000 00mm, bc=mmmm eeee errr rrrr
165C 21210019           lxi h,ranrec! dad d ; hl = .fcb(ranrec)
1660 7123702377         mov m,c! inx h! mov m,b! inx h! mov m,a ; to ranrec
1665 C9                 ret

                disk$select:
                        ; Select disk info for subsequent input or output ops
1666 32CC1C             sta adrive
                disk$select1: ; called by deblock
1669 77                 mov m,a ; curdsk = seldsk or adrive
166A 57                 mov d,a ; Save seldsk in register D for selectdisk call
166B 2A881CCDDB         lhld dlog! call test$vector ; test$vector does not modify DE
1671 5FD5               mov e,a! push d ; Send to seldsk, save for test below
1673 CDFF07E1           call selectdisk! pop h ; Recall dlog vector
1677 D28607             jnc sel$error ; returns with C flag set if select ok
                        ; Is the disk logged in?
167A 2D                 dcr l ; reg l = 1 if so
167B C9                 ret

                tmpselect:
167C 21C61C73           lxi h,seldsk! mov m,e

                curselect:
1680 3AC61C21CB         lda seldsk! lxi h,curdsk! cmp m! jnz select
168A FEFFC0             cpi 0ffh! rnz ; return if seldsk ~= ffh

                select:
168D CD6616             call disk$select

                if MPM
                        jnz select1 ; no
                        ; yes - drive previously logged in
                        lhld rlog! call test$vector
                        sta rem$drv! ret ; Set rem$drv & return
                select1:
```

CP/M - PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3 - 135

```

                 else
1690 C8                 rz ; yes - drive previously logged in
                 endif

1691 CD900D             call initialize ; Log in the directory

                        ; Increment login sequence # if odd
1694 2A961C7EE6         lhld lsn$add! mov a,m! ani 1! push a! add m! mov m,a
169D F1C4A60B           pop a! cnz set$rlog

16A1 CDC30A             call set$dlog

                 if MPM
                        lxi h,chksiz+1! mov a,m! ral! mvi a,0! jc select2
                        lxi d,rlog! call set$cdisk ; rlog=set$cdisk(rlog)
                        mvi a,1
                 select2:
                        sta rem$drv
                 endif

16A4 C9                 ret

                 reselectx:
16A5 AF32D71C           xra a! sta high$ext

                 if BANKED
                        sta xfcb$read$only
                 endif

16A9 C3C316             jmp reselect1

                 reselect:
                        ; Check current fcb to see if reselection necessary
16AC 017F80             lxi b,807fh
16AF 2ADB1E1107         lhld info! lxi d,7! xchg! dad d

                 if BANKED
                        ; xfcb$read$only = 80h & fcb(7)
                        mov a,m! ana b! sta xfcb$read$only
                        ; fcb(7) = fcb(7) & 7fh
                        mov a,m! ana c! mov m,a
                 endif

                 if MPM
                        ; if fcb(8) & 80h
                        ;    then fcb(8) = fcb(8) & 7fh, high$ext = 60h
                        ;    else high$ext = fcb(ext) & 0e0h
                        inx h! lxi d,4
                        mov a,m! ana c! cmp m! mov m,a! mvi a,60h! jnz reselect0
                        dad d! mvi a,0e0h! ana m
                 reselect0:
                        sta high$ext
                 else
                        ; high$ext = 80h & fcb(8)
16B7 237EA032D7         inx h! mov a,m! ana b! sta high$ext
```

```
                          ; fcb(8) = fcb(8) & 7fh
16BD 7EA177               mov a,m! ana c! mov m,a
                endif

                          ; fcb(ext) = fcb(ext) & 1fh
16C0 CD180B               call clr$ext
                reselect1:

16C3 210000               lxi h,0

                if BANKED
                          shld make$xfcb ; make$xfcb,find$xfcb = 0
                endif
16C6 22DF1C               shld xdcnt ; required by directory hashing

16C9 AF32E11C             xra a! sta search$user0
16CD 3D32DD1E             dcr a! sta resel ; Mark possible reselect
16D1 2ADB1E7E             lhld info! mov a,m ; drive select code
16D5 32E31C               sta fcbdsk ; save drive code
16D8 E61F                 ani 1$1111b ; non zero is auto drive select
16DA 3D                   dcr a ; Drive code normalized to 0..30, or 255
16DB 32C31C               sta linfo ; Save drive code
16DE FEFFCAE616           cpi 0ffh! jz noselect
                                  ; auto select function, seldsk saved above
16E3 32C61C                       sta seldsk
                          noselect:
16E6 CD8016                       call curselect
                                  ; Set user code
16E9 3AE01E                       lda usrcode ; 0...15
16EC 2ADB1E77                     lhld info! mov m,a
                          noselect0:
                                  ; Discard directory BCB's if drive is removable
                                  ; and fx = 15,17,19,22,23,30 etc.
16F0 CDB30BCC16                   call tst$log$fxs! cz discard$dir
                                  ; Check for media change on currently slected disk
16F6 CDFC16                       call check$media
                                  ; Check for media change on any other disks
16F9 C33617                       jmp check$all$media

                check$media:
                          ; Check media if DPH media flag set.
                          ; Is DPH media flag set?
16FC CDCA0BC8             call test$media$flag! rz ; no
                          ; Test for media change by reading directory
                          ; to current high water mark or until media change
                          ; is detected.
                          ; First reset DPH media flag & discard directory BCB's
1700 3600                 mvi m,0
1702 CD160C               call discard$dir
1705 2AE11EE5             lhld dcnt! push h
1709 CD4C08CD9D           call home! call set$end$dir
                check$media1:
170F 0E00CDAA0C           mvi c,false! call r$dir
1714 21DE1E7EB7           lxi h,relog! mov a,m! ora a! jz check$media2
171C 3600E13ADF           mvi m,0! pop h! lda fx! cpi 48! rz
1725 CDF10BC3D1           call drv$relog! jmp chk$exit$fxs
```

```
                        check$media2:
1728 CD270BDA0F                 call comp$cdr! jc check$media1
1731 E122E11EC9                 pop h! shld dcnt! ret

                        check$all$media:
                                ; This routine checks all logged-in drives for
                                ; a set DPH media flag and pending buffers.  It reads
                                ; the directory for these drives to verify that media
                                ; has not changed.  If media has changed, the drives
                                ; get reset (but not relogged-in).
                                ; Is SCB media flag set?
1736 21F01E7EB7                 lxi h,media$flag! mov a,m! ora a! rz ; no
                                ; Reset SCB media flag
173C 3600                       mvi m,0
                                ; Test logged-in drives only
173E 2A881C3E10                 lhld dlog! mvi a,16
                        chK$am1:
1743 3D29D25817                 dcr a! dad h! jnc chK$am2
                                ; A = drive #
                                ; Select drive
1748 F5E521CB1C                 push a! push h! lxi h,curdsk! call disk$select
                                ; Does drive have pending data buffers?
1750 CD5F17C4FC                 call test$pending! cnz check$media ; yes
1756 E1F1                       pop h! pop a
                        chK$am2:
1758 B7C24317                   ora a! jnz chK$am1
175C C38016                     jmp curselect

                        test$pending:
                                ; On return, Z flag reset if buffer pending

                                ; Does dta$bcba = 0ffffh
175F 2AA01C7DA4                 lhld dta$bcba! mov a,l! ana h! inr a! rz ; yes

                        if BANKED

                        test$p1:
                                ; Does bcb addr = 0?
                                mov e,m! inx h! mov d,m
                                mov a,e! ora d! rz ; yes - no pending buffers
                                lxi h,4
                        else
1766 110400                     lxi d,4
                        endif

                                ; Is buffer pending?
1769 197EB7                     dad d! mov a,m! ora a ; A ^= 0 if so

                        if BANKED
                                rnz ; yes
                                ; no - advance to next bcb
                                lxi h,13! dad d! jmp test$p1
                        else
176C C9                         ret
                        endif

```

```
                   get$dir$mode:
176D 2A941C7E              lhld drvlbla! mov a,m

                   if not BANKED
1771 E67F                  ani 7fh ; Mask off password bit
                   endif

1773 C9                    ret

                   if BANKED

                   chk$password:
                           call get$dir$mode! ani 80h! rz

                   chk$pw:                 ; Check password
                           call get$xfcb! rz ; a = xfcb options
                           jmp cmp$pw

                   chk$pw$error:
                           ; Disable special searches
                           xra a! sta xdcnt+1
                           ; pw$fcb = dir$xfcb
                           call getdptra! xchg
                           mvi c,12! lxi h,pw$fcb! push h
                           call move! ldax d! inx h! mov m,a! pop d
                           lhld info! mov a,m! stax d
                           ; push original info and xfcb password mode
                           ; info = .pw$fcb
                           push h! xchg! shld info
                           ; Does fcb(ext = 0, mod = 0) exist?
                           call search$namlen! jz chk$pwe2 ; no
                           ; Does sfcb exist for fcb ?
                           call get$dtba$8! ora a! jnz chk$pwe1 ; no
                           xchg! lxi h,pw$mode
                           ; Is sfcb password mode nonzero?
                           mov b,m! ldax d! mov m,a! ora a! jz chk$pwe2 ; no
                           ; Do password modes match?
                           xra b! ani 0e0h! jz chk$pwe1 ; yes
                           ; no - update xfcb to match sfcb
                           call get$xfcb! jz chk$pwe1 ; no xfcb (error)
                           lda pw$mode! mov m,a! call nowrite! cz seek$copy
                   chk$pwe1:
                           pop h! shld info
                           lda fx! cpi 15! rz! cpi 22! rz

                   pw$error:       ; password error
                           mvi a,7! jmp set$aret

                   chk$pwe2:
                           xra a! sta pw$mode
                           call nowrite! jnz chk$pwe3
                           ; Delete xfcb
                           call get$xfcb! push a
                           lhld info! mov a,m! ori 10h! mov m,a
                           pop a! cnz delete$10
                   chk$pwe3:
```

```
                ; Restore info
                pop h! shld info! ret

        cmp$pw:                 ; Compare passwords
                inx h! mov b,m
                mov a,b! ora a! jnz cmp$pw2
                mov d,h! mov e,l! inx h! inx h
                mvi c,9
        cmp$pw1:
                inx h! mov a,m! dcr c! rz
                ora a! jz cmp$pw1
                cpi 20h! jz cmp$pw1
                xchg
        cmp$pw2:
                lxi d,(23-ubytes)! dad d! xchg
                lhld xdmaad! mvi c,8
        cmp$pw3:
                ldax d! xra b! cmp m! jnz cmp$pw4
                dcx d! inx h! dcr c! jnz cmp$pw3
                ret
        cmp$pw4:
                dcx d! dcr c! jnz cmp$pw4
                inx d

        if MPM
                call get$df$pwa! inr a! jnz cmp$pw5
                inr a! ret
        cmp$pw5:

        else
                lxi h,df$password
        endif

                mvi c,8! jmp compare

        if MPM

        get$df$pwa:     ; a = ff => no df pwa
                call rlr! lxi b,console! dad b
                mov a,m! cpi 16! mvi a,0ffh! rnc
                mov a,m! add a! add a! add a
                mvi h,0! mov l,a! lxi b,dfpassword! dad b
                ret
        endif

        set$pw:                 ; Set password in xfcb
                push h ; Save .xfcb(ex)
                lxi b,8 ; b = 0, c = 8
                lxi d,(23-extnum)! dad d
                xchg! lhld xdmaad
        set$pw0:
                xra a! push a
        set$pw1:
                mov a,m! stax d! ora a! jz set$pw2
                cpi 20h! jz set$pw2
                inx sp! inx sp! push a
```

```
                        set$pw2:
                                add b! mov b,a
                                dcx d! inx h! dcr c! jnz set$pw1
                                pop a! ora b! pop h! jnz set$pw3
                                ; is fx = 100 (directory label)?
                                lda fx! cpi 100! jz set$pw3 ; yes
                                mvi m,0 ; zero xfcb(ex) - no password
                        set$pw3:
                                inx d! mvi c,8
                        set$pw4:
                                ldax d! xra b! stax d! inx d! dcr c! jnz set$pw4
                                inx h! ret

                        get$xfcb:
                                lhld info! mov a,m! push a
                                ori 010h! mov m,a
                                call search$extnum! mvi a,0! sta lret
                                lhld info! pop b! mov m,b! rz
                        get$xfcb1:
                                call getdptra! xchg
                                lxi h,extnum! dad d! mov a,m! ani 0e0h! ori 1
                                ret

                        adjust$dmaad:
                                push h! lhld xdmaad! dad d
                                shld xdmaad! pop h! ret

                        init$xfcb:
                                call setcdr ; may have extended the directory
                                lxi b,1014h ; b=10h, c=20
                        init$xfcb0:
                                ; b = fcb(0) logical or mask
                                ; c = zero count
                                push b
                                call getdptra! xchg! lhld info! xchg
                                ; Zero extnum and modnum
                                ldax d! ora b! mov m,a! inx d! inx h
                                mvi c,11! call move! pop b! inr c
                        init$xfcb1:
                                dcr c! rz
                                mvi m,0! inx h! jmp init$xfcb1

                        chk$xfcb$password:
                                call get$xfcb1
                        chk$xfcb$password1:
                                push h! call cmp$pw! pop h! ret

                        endif

                        stamp1:
1774 0E00C37B17                 mvi c,0! jmp stamp3
                        stamp2:
1779 0E04                       mvi c,4
                        stamp3:
177B CDA717B7C0                 call get$dtba! ora a! rnz
1780 113E0CD5                   lxi d,seek$copy! push d
```

```
                        stamp4:

                        if MPM
                                  push h
                                  call get$stamp$add! xchg
                                  pop h
                        else
1784 11F41E                       lxi d,stamp
                        endif

1787 E5D5                         push h! push d
1789 0E00CD4EFF                   mvi c,0! call timef ; does not modify hl,de
178E 0E04CDEB07                   mvi c,4! call compare
1793 0E04D1E1C2                   mvi c,4! pop d! pop h! jnz move
179A E1C9                         pop h! ret

                        stamp5:
179C CDFE0A0911                   call getdptra! dad b! lxi d,func$ret! push d
17A4 C38417                       jmp stamp4

                        if BANKED

                        get$dtba$8:
                                  mvi c,8
                        endif

                        get$dtba:
                                  ; c = offset of sfcb subfield (0,4,8)
                                  ; Return with a = 0 if sfcb exists

                                  ; Does fcb occupy 4th item of sector?
17A7 3AE11EE603                   lda dcnt! ani 3! cpi 3! rz ; yes
17AF 47                           mov b,a
17B0 2A8C1C1160                   lhld buffa! lxi d,96! dad d
                                  ; Does sfcb reside in 4th directory item?
17B7 7ED621C0                     mov a,m! sui 21h! rnz ; no
                                  ; hl = hl + 10*lret + 1 + c
17BB 78875F8787                   mov a,b! add a! mov e,a! add a! add a! add e
17C1 3C815F19AF                   inr a! add c! mov e,a! dad d! xra a
17C6 C9                           ret

                        qstamp:
                                  ; Is fcb 1st logical fcb for file?
17C7 CDD517C0                     call qdirfcb1! rnz ; no
                        qstamp1:
                                  ; Does directory label specify requested stamp?
17CB 2A941C79A6                   lhld drvlbla! mov a,c! ana m! jnz nowrite ; yes - verify drive r/w
17D3 3CC9                         inr a! ret ; no - return with Z flag reset

                        qdirfcb1:
                                  ; Routine to determine if fcb is 1st directory fcb
                                  ; for file
                                  ; Is fcb(ext) & ~extmsk & 00011111b = 0?
17D5 3AA91CF6E0                   lda extmsk! ori 1110$0000b! cma! mov b,a
17DC CDFD097EA0                   call getexta! mov a,m! ana b! rnz ; no
                                  ; is fcb(mod) & 0011$1111B = 0?
```

```
17E2 23237EE63F         inx h! inx h! mov a,m! ani 3fh! ret ; Z flag set if zero

                update$stamp:
                        ; Is update stamping requested on drive?
17E8 0E20CDCB17         mvi c,0010$0000b! call qstamp1! rnz ; no
                        ; Has file been written to since it was opened?
17EE CD090BE640         call getmodnum! ani 40h! rnz ; yes - update stamp performed
                        ; Search for 1st dir fcb
17F4 CDFD094636         call getexta! mov b,m! mvi m,0! push h
17FB 23234E3600         inx h! inx h! mov c,m! mvi m,0! push b
                        ; Search from beginning of directory
1801 CD860E             call search$namlen
                        ; Perform update stamp if dir fcb 1 found
1804 C47917             cnz stamp2
1807 AF327A07           xra a! sta lret
                        ; Restore fcb extent and module fields
180B C1E1702323         pop b! pop h! mov m,b! inx h! inx h! mov m,c! ret

                if MPM

                pack$sdcnt:

                ;packed$dcnt = dblk(low 15 bits) || dcnt(low 9 bits)

                ;       if sdblk = 0 then dblk = shr(sdcnt,blkshf+2)
                ;                    else dblk = sdblk
                ;       dcnt = sdcnt & (blkmsk || '11'b)
                ;
                ;       packed$dcnt format (24 bits)
                ;
                ;       12345678 12345678 12345678
                ;       23456789 .......1 ........ sdcnt (low 9 bits)
                ;       ........ 9abcdef. 12345678 sdblk (low 15 bits)
                ;
                        lhld sdblk! mov a,h! ora l! jnz pack$sdcnt1
                        lda blkshf! adi 2! mov c,a! lhld sdcnt
                        call hlrotr
                pack$sdcnt1:
                        dad h! xchg! lxi h,sdcnt! mvi b,1
                        lda blkmsk! ral! ora b! ral! ora b
                        ana m! sta packed$dcnt
                        lda blkshf! cpi 7! jnz pack$sdcnt2
                        inx h! mov a,m! ana b! jz pack$sdcnt2
                        mov a,e! ora b! mov e,a
                pack$sdcnt2:
                        xchg! shld packed$dcnt+1
                        ret

                ; olist element = link(2) || atts(1) || dcnt(3) ||
                ;                pdaddr(2) || opncnt(2)
                ;
                ;       link = 0 -> end of list
                ;
                ;       atts - 80 - open in locked mode
                ;              40 - open in unlocked mode
                ;              20 - open in read/only mode
```

```
        ;               10 - deleted item
        ;               0n - drive code (0-f)
        ;
        ;       dcnt = packed sdcnt+sdblk
        ;       pdaddr = process descriptor addr
        ;       opncnt = # of open calls - # of close calls
        ;                olist item freed by close when opncnt = 0
        ;
        ; llist element = link(2) ¦¦ drive(1) ¦¦ arecord(3) ¦¦
        ;                pdaddr(2) ¦¦ .olist$item(2)
        ;
        ;       link = 0 -> end of list
        ;
        ;       drive - 0n - drive code (0-f)
        ;
        ;       arecord = record number of locked record
        ;       pdaddr = process descriptor addr
        ;       .olist$item = address of file's olist item

        search$olist:
                lxi h,open$root! jmp srch$list0
        search$llist:
                lxi h,lock$root! jmp srch$list0
        searchn$list:
                lhld cur$pos
        srch$list0:
                shld prv$pos

        ; search$olist, search$llist, searchn$list conventions
        ;
        ;       b = 0 -> return next item
        ;       b = 1 -> search for matching drive
        ;       b = 3 -> search for matching dcnt
        ;       b = 5 -> search for matching dcnt + pdaddr
        ;       if found then z flag is set
        ;                       prv$pos -> previous list element
        ;                       cur$pos -> found list element
        ;                       hl -> found list element
        ;       else prv$pos -> list element to insert after
        ;
        ;       olist and llist are maintained in drive order

        srch$list1:
                mov e,m! inx h! mov d,m! xchg
                mov a,l! ora h! jz srch$list3
                xra a! cmp b! jz srch$list6
                inx h! inx h!
                lxi d,curdsk! mov a,m! ani 0fh! mov c,a
                ldax d! sub c! jnz srch$list4
                mov a,b! dcr a! jz srch$list5
                mov c,b! push h
                inx d! inx h! call compare
                pop h! jz srch$list5
        srch$list2:
                dcx h! dcx h
                shld prv$pos! jmp srch$list1
```

```
srch$list3:
        inr a! ret
srch$list4:
        jnc srch$list2
srch$list5:
        dcx h! dcx h
srch$list6:
        shld cur$pos! ret

delete$item: ; hl -> item to be deleted
        di
        push d! push h
        mov e,m! inx h! mov d,m
        lhld prv$pos! shld cur$pos
        ; prv$pos.link = delete$item.link
        mov m,e! inx h! mov m,d

        lhld free$root! xchg
        ; free$root = .delete$item
        pop h! shld free$root
        ; delete$item.link = previous free$root
        mov m,e! inx h! mov m,d
        pop d! ei! ret

create$item: ; hl -> new item if successful
                ; z flag set if no free items
        lhld free$root! mov a,l! ora h! rz
        push d! push h! shld cur$pos
        mov e,m! inx h! mov d,m
        ; free$root = free$root.link
        xchg! shld free$root

        lhld prv$pos
        mov e,m! inx h! mov d,m
        pop h
        ; create$item.link = prv$pos.link
        mov m,e! inx h! mov m,d! dcx h
        xchg! lhld prv$pos
        ; prv$pos.link = .create$item
        mov m,e! inx h! mov m,d! xchg
        pop d! ret

set$olist$item:
        ; a = attributes
        ; hl = olist entry address
        inx h! inx h
        mov b,a! lxi d,curdsk! ldax d! ora b
        mov m,a! inx h! inx d
        mvi c,5! call move
        xra a! mov m,a! inx h! mov m,a! ret

set$sdcnt:
        mvi a,0ffh! sta sdcnt+1! ret

tst$olist:
        mvi a,0c9h! sta chk$olist05! jmp chk$olist0
```

```
chK$olist:
       xra a! sta chK$olist05
chK$olist0:
       lxi d,dcnt! lxi h,sdcnt! mvi c,4! call move
       call pack$sdcnt! mvi b,3! call search$olist! rnz
       pop d ; pop return address
       inx h! inx h
       mov a,m! ani 80h! jz openx06
       dcx h! dcx h
       push d! push h
       call compare$pds! pop h! pop d! jnz openx06
       push d ; Restore return address
chK$olist05:
       nop ; tst$olist changes this instr to ret
       call delete$item! lda pdcnt
chK$olist1:
       adi 16! jz chK$olist1
       sta pdcnt

       push a! call rlr
       lxi b,pdcnt$off! dad b! pop a
       mov m,a! ret

remove$files:  ; bc = pdaddr
       lhld cur$pos! push h
       lhld prv$pos! push h
       mov d,b! mov e,c! lxi h,open$root! shld cur$pos
remove$file1:
       mvi b,0! push d! call searchn$list! pop d! jnz remove$file2
       lxi b,6! call tst$tbl$lmt! jnz remove$file1
       inx h! inx h! mov a,m! ori 10h! mov m,a
       sta deleted$files
       jmp remove$file1
remove$file2:
       pop h! shld prv$pos
       pop h! shld cur$pos
       ret

delete$files:
       lxi h,open$root! shld cur$pos
delete$file1:
       mvi b,0! call search$nlist! rnz
       inx h! inx h! mov a,m! ani 10h! jz delete$file1
       dcx h! dcx h! call remove$locks! call delete$item
       jmp delete$file1

flush$files:
       lxi h,flushed! mov a,m! ora a! rnz
       inr m
flush$file0:
       lxi h,open$root! shld cur$pos
flush$file1:
       mvi b,1! call searchn$list! rnz
       push h! call remove$locks! call delete$item! pop h
       lxi d,6! dad d! mov e,m! inx h! mov d,m
       lxi h,pdcnt$off! dad d! mov a,m! ani 1! jnz flush$file1
```

```
		mov a,m! ori 1! mov m,a
		lhld pdaddr! mvi c,2! call compare! jnz flush$file1
		lda pdcnt! adi 10h! sta pdcnt! jmp flush$file1

	free$files:
		; free$mode = 1 - remove curdsk files for process
		;             0 - remove all files for process
		lhld pdaddr! xchg! lxi h,open$root! shld curpos
	free$files1:
		lda free$mode! mov b,a
		push d! call searchn$list! pop d! rnz
		lxi b,6! call tst$tbl$lmt! jnz free$files1
		push h! inx h! inx h! inx h
		call test$ffff! jnz free$files2
		call test$ffff! jz free$files3
	free$files2:
		mvi a,0ffh! sta incr$pdcnt
	free$files3:
		pop h! call remove$locks! call delete$item
		jmp free$files1

	remove$locks:
		shld file$id
		inx h! inx h! mov a,m! ani 40h! jz remove$lock3
		push d! lhld prv$pos! push h
		lhld file$id! xchg! lxi h,lock$root! shld cur$pos
	remove$lock1:
		mvi b,0! push d! call searchn$list! pop d
		jnz remove$lock2
		lxi b,8! call tst$tbl$lmt! jnz remove$lock1
		call delete$item
		jmp remove$lock1
	remove$lock2:
		pop h! shld prv$pos! pop d
	remove$lock3:
		lhld file$id! ret

	tst$tbl$lmt:
		push h! dad b
		mov a,m! inx h! mov h,m
		sub e! jnz tst$tbl$lmt1
		mov a,h! sub d
	tst$tbl$lmt1:
		pop h! ret

	create$olist$item:
		mvi b,1! call search$olist
		di
		call create$item! lda attributes! call set$olist$item
		ei
		ret

	count$opens:
		xra a! sta open$cnt
		lhld pdaddr! xchg! lxi h,open$root! shld curpos
	count$open1:
        mvi b,0! push d! call searchn$list! pop d! jnz count$open2
        lxi b,6! call tst$tbl$lmt! jnz count$open1
        lda open$cnt! inr a! sta open$cnt
        jmp count$open1
count$open2:
        lxi h,open$max! lda open$cnt! ret

count$locks:
        xra a! sta lock$cnt
        xchg! lxi h,lock$root! shld cur$pos
count$lock1:
        mvi b,0! push d! call searchn$list! pop d! rnz
        lxi b,8! call tst$tbl$lmt! jnz count$lock1
        lda lock$cnt! inr a! sta lock$cnt
        jmp count$lock1

check$free:
        lda mult$cnt! mov e,a
        mvi d,0! lxi h,free$root! shld cur$pos
check$free1:
        mvi b,0! push d! call searchn$list! pop d! jnz check$free2
        inr d! mov a,d! sub e! jc check$free1
        ret
check$free2:
        pop h! mvi a,14! jmp sta$ret

lock:                           ; record lock and unlock
        call reselect! call check$fcb
        call test$unlocked
        rz ; file not opened in unlocked mode
        lhld xdmaad! mov e,m! inx h! mov d,m
        xchg! inx h! inx h
        mov a,m! mov b,a! lda curdsk! sub b
        ani 0fh! jnz lock8 ; invalid file id
        mov a,b! ani 40h! jz lock8 ; invalid file id
        dcx h! dcx h! shld file$id
        lda lock$unlock! inr a! jnz lock1 ; jmp if unlock
        call count$locks
        lda lock$cnt! mov b,a
        lda mult$cnt! add b! mov b,a
        lda lock$max! cmp b
        mvi a,12! jc sta$ret ; too many locks by this process
        call check$free
lock1:
        call save$rr! lxi h,lock9! push h! lda mult$cnt
lock2:
        push a! call get$lock$add
        lda lock$unlock! inr a! jnz lock3
        call test$lock
lock3:
        pop a! dcr a! jz lock4
        call incr$rr! jmp lock2
lock4:
        call reset$rr! lda mult$cnt
lock5:
        push a! call get$lock$add
```

```
		lda lock$unlock! inr a! jnz lock6
		call set$lock! jmp lock7
	lock6:
		call free$lock
	lock7:
		pop a! dcr a! rz
		call incr$rr! jmp lock5
	lock8:
		mvi a,13! jmp sta$ret ; invalid file id
	lock9:
		call reset$rr! ret

	get$lock$add:
		lxi h,0! dad sp! shld lock$sp
		mvi a,0ffh! sta lock$shell
		call rseek
		xra a! sta lock$shell
		call getfcb
		lhld aret! mov a,l! ora a! jnz lock$err
		call index! lxi h,1! jz lock$err
		call atran! ret

	lock$perr:
		xra a! sta lock$shell
		xchg! lhld lock$sp! sphl! xchg
	lock$err:
		pop d ; Discard return address
		pop b ; b = mult$cnt-# recs processed
		lda mult$cnt! sub b
		add a! add a! add a! add a
		ora h! mov h,a! mov b,a
		shld aret! ret

	test$lock:
		call move$arecord
		mvi b,3! call search$llist! rnz
		call compare$pds! rz
		lxi h,8! jmp lock$err

	set$lock:
		call move$arecord
		mvi b,1! call search$llist
		di
		call create$item
		xra a! call set$alist$item
		xchg! lhld file$id! xchg
		mov m,d! dcx h! mov m,e
		ei! ret

	free$lock:
		call move$arecord
		mvi b,5! call search$llist! rnz
	free$lock0:
		call delete$item
		mvi b,5! call searchn$list! rnz
		jmp free$lock0
```

```

compare$pds:
        lxi d,6! dad d! xchg
        lxi h,pdaddr! mvi c,2! jmp compare


move$arecord:
        lxi d,arecord! lxi h,packed$dcnt


fix$olist$item:
        lxi d,xdcnt! lxi h,sdcnt
        ; Is xdblk,xdcnt < sdblk,sdcnt
        mvi c,4! ora a!
fix$ol1:
        ldax d! sbb m! inx h! inx d! dcr c! jnz fix$ol1
        rnc
        ; yes - update olist entry
        call swap! call sdcnt$eq$xdcnt
        lxi h,open$root! shld cur$pos
        ; Find file's olist entry
fix$ol2:
        call swap! call pack$sdcnt! call swap
        mvi b,3! call searchn$list! rnz
        ; Update olist entry with new dcnt value
        push h! call pack$sdcnt! pop h
        inx h! inx h! inx h! lxi d,packed$dcnt
        mvi c,3! call move! jmp fix$ol2

hl$eq$hl$and$de:
        mov a,l! ana e! mov l,a
        mov a,h! ana d! mov h,a
        ret

remove$drive:
        xchg! lda curdsk! mov c,a! lxi h,1
        call hlrotl
        mov a,l! cma! ana e! mov e,a
        mov a,h! cma! ana d! mov d,a
        xchg! ret

diskreset:
        lxi h,0! shld ntlog
        xra a! sta set$ro$flag
        lhld info
intrnldiskreset:
        xchg! lhld open$root! mov a,h! ora l! rz
        xchg! lda curdsk! push a! mvi b,0
dskrst1:
        mov a,l! rar! jc dskrst3
dskrst2:
        mvi c,1! call hlrotr! inr b
        mov a,h! ora l! jnz dskrst1
        pop a! sta curdsk
        lhld ntlog! xchg! lhld tlog
        mov a,l! ora e! mov l,a
        mov a,h! ora d! mov h,a! shld tlog
        inr a! ret
dskrst3:
        push b! push h! mov a,b! sta curdsk
        lhld rlog! call test$vector1! push a
        lhld rodsk! lda curdsk! call test$vector1! mov b,a
        pop h! lda set$ro$flag! ora b! ora h! sta check$disk
        lxi h,open$root! shld cur$pos
dskrst4:
        mvi b,1! call searchn$list! jnz dskrst6
        lda check$disk! ora a! jz dskrst5
        push h! call compare$pds! jz dskrst45
        pop h! xra a! xchg! jmp dskrst6
dskrst45:
        lxi d,ntlog! call set$cdisk
        pop h! jmp dskrst4
dskrst5:
        lhld info! call remove$drive! shld info
        ori 1
dskrst6:
        pop h! pop b! jnz dskrst2

        ; error - olist item exists for another process
        ; for removable drive to be reset
        pop a! sta curdsk! mov a,b! adi 41h ; a = ascii drive
        lxi h,6! dad d! mov c,m! inx h! mov b,m ; bc = pdaddr
        push psw! call test$error$mode! pop d! jnz dskrst7
        mov a,d

        push b! push psw
        call rlr! lxi d,console! dad d! mov d,m ; d = console #
        lxi b,deniedmsg! call xprint
        pop psw! mov c,a! call conoutx
        mvi c,':'! call conoutx
        lxi b,cnsmsg! call xprint
        pop h! push h! lxi b,console! dad b
        mov a,m! adi '0'! mov c,a! call conoutx
        lxi b,progmsg! call xprint
        pop h! call dsplynm

dskrst7:
        pop h ; Remove return addr from diskreset
        lxi h,0ffffh! shld aret ; Flag the error
        ret

deniedmsg:
        db cr,lf,'disk reset denied, drive ',0
cnsmsg:
        db ' console ',0
progmsg:
        db ' program ',0
endif

;
;       individual function handlers
;
```

```

                func12:
                        ; Return version number

                if MPM
                        lxi h,0100h+dvers! jmp sthl$ret
                else
1812 3AA11EC3CB         lda version! jmp sta$ret ; lret = dvers (high = 00)
                endif

                func13:

                if MPM
                        lhld dlog! shld info
                        call diskreset! jz reset$all
                        call reset$37
                        jmp func13$cont
                reset$all:

                        ; Reset disk system - initialize to disk 0
                        lxi h,0! shld rodsk! shld dlog

                        shld rlog! shld tlog
                func13$cont:
                        mvi a,0ffh! sta curdsk
                else
1818 21FFFFCDF4         lxi h,0ffffh! call reset$37x
                endif
181E AF32DA1E           xra a! sta olddsk ; Note that usrcode remains unchanged

                if MPM
                        xra a! call getmemseg ; a = mem seg tbl index
                        ora a! rz
                        inr a! rz
                        call rlradr! lxi b,mseqtbl-rlros! dad b
                        add a! add a! mov e,a! mvi d,0! dad d
                        mov h,m! mvi l,80h
                        jmp intrnlsetdma
                else
1822 21800022D8         lxi h,tbuff! shld dmaad ; dmaad = tbuff
1828 C3520C              jmp setdata ; to data dma address
                endif

                func14:

                if MPM
                        call tmpselect ; seldsk = reg e
                        call rlr! lxi b,diskselect! dad b
                        mov a,m! ani 0fh! rrc! rrc! rrc! rrc
                        mov b,a! lda seldsk! ora b! rrc! rrc! rrc! rrc
                        mov m,a! ret
                else
182B CD7C16             call tmpselect ; seldsk = reg e
182E 3AC61C32DA         lda seldsk! sta olddsk! ret
                endif

```

```
                      func15:
                              ; Open file
 1835 CD120B                  call clrmodnum ; Clear the module number

                      if MPM
                              call reselect
                              xra a! sta make$flag
                              call set$sdcnt
                              lxi h,open$file! push h
                              mvi a,0c9h! sta check$fcb4
                              call check$fcb1
                              pop h! lda high$ext! cpi 060h! jnz open$file
                              call home! call set$end$dir
                              jmp open$user$zero
                      open$file:
                              call set$sdcnt
                              call reset$chksum$fcb ; Set invalid check sum
                      else
 1838 CDA516                  call reselectx
                      endif

 183B CD1510                  call check$wild ; Check for wild chars in fcb

                      if MPM

                              call get$atts! ani 1100$0000b ; a = attributes
                              cpi 1100$0000b! jnz att$ok
                              ani 0100$0000b ; Mask off unlock mode
                      att$ok:
                              sta high$ext
                              mov b,a! ora a! rar! jnz att$set
                              mvi a,80h
                      att$set:
                              sta attributes! mov a,b
                              ani 80h! jnz call$open
                      endif

 183E 3AE01EB7CA              lda usrcode! ora a! jz call$open
 1845 3EFE32E01C              mvi a,0feh! sta xdcnt+1! inr a! sta search$user0

                      if MPM
                              sta sdcnt0+1
                      endif

                      call$open:
 184E CDA110CD7C              call open! call openx ; returns if unsuccessful, a = 0
 1854 21E11CBEC8              lxi h,search$user0! cmp m! rz
 1859 773AE01CFE              mov m,a! lda xdcnt+1! cpi 0feh! rz
                      ;
                      ;       file exists under user 0
                      ;

                      if MPM
                              call swap
                      endif

```

```
1860 CDD111             call set$dcnt$dblk

                if MPM
                        mvi a,0110$0000b
                else
1863 3E80               mvi a,80h
                endif

1865 32D71C             sta high$ext
                open$user$zero:
                        ; Set fcb user # to zero
1868 2ADB1E3600         lhld info! mvi m,0
186D 0E0FCD730E         mvi c,namlen! call searchi! call searchn
1875 CDA410             call open1 ; Attempt reopen under user zero
1878 CD7C18             call openx ; openx returns only if unsuccessful
187B C9                 ret
                openx:
187C CD940CC8           call end$of$dir! rz
1880 CD0D0A7E3C         call getfcba! mov a,m! inr a! jnz openxa
1888 1B1B1A77           dcx d! dcx d! ldax d! mov m,a
                openxa:
                        ; open successful
188C E1                 pop h ; Discard return address
                        ; Was file opened under user 0 after unsuccessful
                        ; attempt to open under user n?

                if MPM
                        lda high$ext! cpi 060h! jz openx00 ; yes
                        ; Was file opened in locked mode?
                        ora a! jnz openx0 ; no
                        ; does user = zero?
                        lhld info! ora m! jnz openx0 ; no
                        ; Does file have read/only attribute set?
                        call rotest! jnc openx0 ; no
                        ; Does file have system attribute set?
                        inx h! mov a,m! ral! jnc openx0 ; no

                        ; Force open mode to read/only mode and set user 0 flag
                        ; if file opened in locked mode, user = 0, and
                        ; file has read/only and system attributes set

                openx00:

                else
188D 3AD71C17D2         lda high$ext! ral! jnc openx0
                endif

                        ; Is file under user 0 a system file ?

                if MPM
                        mvi a,20h! sta attributes
                endif

1894 2ADB1E110A         lhld info! lxi d,10! dad d
189B 7EE680C2A7         mov a,m! ani 80h! jnz openx0 ; yes - open successful
                        ; open fails
```

```
18A1 32D71CC326         sta high$ext! jmp lret$eq$ff
                openx0:

                if MPM
                        call reset$chksum$fcb
                else
18A7 CDFF0B             call set$lsn
                endif

                if BANKED

                        ; Are passwords enabled on drive?
                        call get$dir$mode! ani 80h! jz openx1a ; no
                        ; Is this 1st dir fcb?
                        call qdirfcb1! jnz openx0a ; no
                        ; Does sfcb exist?
                        call get$dtba$8! ora a! jnz openx0a ; no
                        ; Is sfcb password mode read or write?
                        mov a,m! ani 0c0h! jz openx1a ; no
                        ; Does xfcb exist?
                        call xdcnt$eq$dcnt
                        call get$xfcb! jnz openx0b ; yes
                        ; no - set sfcb password mode to zero
                        call restore$dir$fcb! rz ; (error)
                        ; Does sfcb still exist?
                        call get$dtba$8! ora a! jnz openx1a ; no (error)
                        ; sfcb password mode = 0
                        mov m,a
                        ; update sfcb
                        call nowrite! cz seek$copy
                        jmp openx1a
                openx0a:
                        call xdcnt$eq$dcnt
                        ; Does xfcb exist?
                        call get$xfcb! jz openx1 ; no
                openx0b:
                        ; yes - check password
                        call cmp$pw! jz openx1
                        call chk$pw$error
                        lda pw$mode! ani 0c0h! jz openx1
                        ani 80h! jnz pw$error
                        mvi a,080h! sta xfcb$read$only
                openx1:
                        call restore$dir$fcb! rz ; (error)
                openx1a:
                        call set$lsn

                if MPM
                        call pack$sdcnt
                        ; Is this file currently open?
                        mvi b,3! call search$olist! jz openx04
                openx01:
                        ; no - is olist full?
                        lhld free$root! mov a,l! ora h! jnz openx03
                        ; yes - error
                openx02:
```

```
                            mvi a,11! jmp set$aret
                    openx03:
                            ; Has process exceeded open file maximum?
                            call count$opens! sub m! jc openx035
                            ; yes - error
                    openx034:
                            mvi a,10! jmp set$aret
                    openx035:
                            ; Create new olist element
                            call create$olist$item
                            jmp openx08
                    openx04:
                            ; Do file attributes match?
                            inx h! inx h
                            lda attributes! ora m! cmp m! jnz openx06
                            ; yes - is open mode locked?
                            ani 80h! jnz openx07
                            ; no - has this file been opened by this process?
                            lhld prv$pos! shld cur$pos
                            mvi b,5! call searchn$list! jnz openx01
                    openx05:
                            ; yes - increment open file count
                            lxi d,8! dad d! inr m! jnz openx08
                            ; count overflow
                            inx h! inr m! jmp openx08
                    openx06:
                            ; error - file opened by another process in incompatible mode
                            mvi a,5! jmp set$aret
                    openx07:
                            ; Does this olist item belong to this process?
                            dcx h! dcx h! push h
                            call compare$pds
                            pop h! jnz openx06 ; no - error
                            jmp openx05 ; yes
                    openx08:; Wopen ok
                            ; Was file opened in unlocked mode?
                            lda attributes! ani 40h! jz openx09 ; no
                            ; yes - return .olist$item in ranrec field of fcb
                            call get$rra
                            lxi d,cur$pos! mvi c,2! call move
                    openx09:
                            call set$fcb$cks$flag
                            lda make$flag! ora a! rnz
                    endif
                    endif

18AA 0E40                   mvi c,0100$0000b
                    openx2:
18AC CDC717CC74             call qstamp! cz stamp1
18B2 11901EC3C6             lxi d,olog! jmp set$cdisk

                    func16:
                            ; Close file
18B8 CDAC16                 call reselect

                    if MPM
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SFR. #

```
                        call get$atts! sta attributes
                        lxi h,close00! push h
                        mvi a,0c9h! sta check$fcb4
                        call check$fcb1! pop h
                        call set$sdcnt
                        call getmodnum! ani 80h! jnz close01
                        call close! jmp close02
                close00:
                        mvi a,6! jmp set$aret
                close01:
                        mvi a,0ffh! sta dont$close! call close1
                close02:
                else
18BB CDFF0B             call set$lsn
18BE CD960ACD04         call cheK$fcb! call close
                endif

18C4 3A7A073CC8         lda lret! inr a! rz

18C9 C3B41A             jmp flush ; Flush buffers

                if MPM
                        lda attributes! ral! rc
                        call pack$sdcnt
                        ; Find olist item for this process & file
                        mvi b,5! call search$olist! jnz close03
                        ; Decrement open count
                        push h! lxi d,8! dad d
                        mov a,m! sui 1! mov m,a! inx h
                        mov a,m! sbi 0! mov m,a! dcx h
                        ; Is open count = 0ffffh
                        call test$ffff! pop h! jnz close03
                        ; yes - remove file's olist entry
                        shld file$id! call delete$item
                        call reset$chksum$fcb
                        ; if unlocked file, remove file's locktbl entries
                        call test$unlocked! jz close03
                        lhld file$id! call remove$locks
                close03:
                        ret

                endif

                func17:
                        ; Search for first occurrence of a file
18CC EBAF               xchg! xra a
                csearch:
18CE F5                 push a
18CF 7EFE3FC2E0         mov a,m! cpi '?'! jnz csearch1 ; no reselect if ?
18D5 CD8016CDF0         call curselect! call noselect0! mvi c,0! jmp csearch3
                csearch1:
18E0 CDFD097EFE         call getexta! mov a,m! cpi '?'! jz csearch2
18E9 CD180BCD12         call clr$ext! call clrmodnum
                csearch2:
18EF CDA516             call reselectx
18F2 0E0F               mvi c,namlen
```

```
                      csearch3:
18F4 F1F5CA0B19               pop a! push a! jz csearch4
                              ; dcnt = dcnt & 0fch
18F9 2AE11EE53E               lhld dcnt! push h! mvi a,0fch
18FF A56F22E11E               ana l! mov l,a! shld dcnt
1904 CD360C                   call rd$dir
1907 E122E11E                 pop h! shld dcnt
                      csearch4:
190B F1                       pop a
190C 215F0C                   lxi h,dir$to$user
190F E5                       push h
1910 CA8D0E                   jz search
1913 3AE51E4FCD               lda searchl! mov c,a! call searchi! jmp searchn

                      func18:
                              ; Search for next occurrence of a file name

                      if BANKED
                              xchg! shld searcha
                      else
191D 2AE31E22DB               lhld searcha! shld info
                      endif

1923 F601C3CE18               ori 1! jmp csearch

                      func19:
                              ; Delete a file
1928 CDA516                   call reselectx
192B C3630F                   jmp delete

                      func20:
                              ; Read a file
192E CDAC16                   call reselect
1931 CD8C0A                   call check$fcb
1934 C3B412                   jmp seqdiskread

                      func21:
                              ; Write a file
1937 CDAC16                   call reselect
193A CD8C0A                   call check$fcb
193D C3AD13                   jmp seqdiskwrite

                      func22:
                              ; Make a file

                      if BANKED
                              call get$atts! sta attributes
                      endif

1940 CD180B                   call clr$ext
1943 CD120B                   call clrmodnum ; fcb mod = 0
1946 CDA516                   call reselectx

                      if MPM
                              call reset$chksum$fcb
                      endif
```

```

1949 CD1510                    call check$wild
194C CDCA11                    call set$xdcnt ; Reset xdcnt for make

                       if MPM
                               call set$sdcnt
                       endif

194F CDA110                    call open ; Verify file does not already exist

                       if MPM
                               call reset$chksum$fcb
                       endif

                               ; Does dir fcb for fcb exist?
                               ; ora a required to reset carry
1952 CD940CB7CA                call end$of$dir! ora a! jz makea0 ; no
                               ; Is dir$ext < fcb(ext)?
1959 CD2D0EBED2                call get$dir$ext! cmp m! jnc file$exists ; no
                       makea0:
1960 F5                        push a ; carry set if dir fcb already exists

                       if MPM
                               lda attributes! ani 80h! rrc! jnz makex00
                               mvi a,80h
                       makex00:
                               sta make$flag
                               lda sdcnt+1! inr a! jz makex01
                               call pack$sdcnt
                               mvi b,3! call search$olist! jz make$x02
                       makex01:
                               lhld free$root! mov a,l! ora h! jz openx02
                               jmp makex03
                       makex02:
                               inx h! inx h
                               lda makeflag! ana m! jz openx06
                               dcx h! dcx h! call compare$pds! jz makex03
                               lda makeflag! ral! jc openx06
                       makex03:

                       endif

                       if BANKED
                               ; Is fcb 1st fcb for file?
                               call qdirfcb1! jz makex04 ; yes
                               ; no - does dir lbl require passwords?
                               call get$dir$mode! ani 80h! jz makex04
                               ; no - does xfcb exist with mode 1 or 2 password?
                               call get$xfcb! jz makex04
                               ; yes - check password
                               call chk$xfcb$password1! jz makex04
                               ; Verify password error
                               call chk$pw$error
                               lda pw$mode! ani 0c0h! jnz pw$error
                       makex04:

                        endif

                                ; carry on stack indicates a make not required because
                                ; of extent folding
1961 F1D4DD11                   pop a! cnc make

                        if MPM
                                call reset$chksum$fcb
                        endif

                                ; end$of$dir call either applies to above make or open call
1965 CD940CC8                   call end$of$dir! rz ; Return if make unsuccessful

                        if not MPM
1969 CDFF0B                     call set$lsn
                        endif

                        if BANKED

                                ; Are passwords activated by dir lbl?
                                call get$dir$mode! ani 80h! jz make3a
                                ; Did user set password attribute?
                                lda attributes! ani 40h! jz make3a
                                ; Is fcb file's 1st logical fcb?
                                call qdirfcb1! jnz make3a
                                ; yes - does xfcb already exist for file
                                call xdcnt$eq$dcnt
                                call get$xfcb! jnz make00 ; yes
                                ; Attempt to make xfcb
                                mvi a,0ffh! sta make$xfcb! call make! jnz make00
                                ; xfcb make failed - delete fcb that was created above
                                call search$namlen
                                call delete10! jmp lret$eq$ff ; Return with a = 0ffh

                        make00:
                                call init$xfcb ; Initialize xfcb
                                ; Get password mode from dma + 8
                                xchg! lhld xdmaad! lxi b,8! dad b! xchg
                                ldax d! ani 0e0h! jnz make2
                                mvi a,080h ; default password mode is read protect
                        make2:
                                sta pw$mode
                                ; Set xfcb password mode field
                                push a! call getxfcb1! pop a! mov m,a
                                ; Set xfcb password and password checksum
                                ; Fix hash table and write xfcb
                                call set$pw! mov m,b! call sdl3
                                ; Return to fcb
                                call restore$dir$fcb! rz
                                ; Does sfcb exist?
                                mvi c,8! call getdtba! ora a! jnz make3a ; no
                                ; Place password mode in sfcb if sfcb exists
                                lda pw$mode! mov m,a! call seek$copy
                                call set$lsn
                        endif
```

```
                      make3a:
196C 0E50                     mvi c,0101$0000b

                      if MPM
                              call openx2
                              lda make$flag! sta attributes
                              ani 40h! ral! sta high$ext
                              lda sdcnt+1! inr a! jnz makexx02
                              call sdcnt$eq$xdcnt! call pack$sdcnt
                              jmp openx03
                      makexx02:
                              call fix$olist$item! jmp openx1
                              jmp set$fcb$cks$flag
                      else
196E CDAC18                   call openx2
1971 0E20CDC717               mvi c,0010$0000b! call qstamp! rnz
1977 CD7917C3A1               call stamp2! jmp set$filewf
                      endif

                      file$exists:
197D 3E08                     mvi a,8
                      set$aret:
197F 4F327B07CD               mov c,a! sta aret+1! call lret$eq$ff

                      if MPM
                              call test$error$mode! jnz goback
                      else
1986 C39307                   jmp goerr1
                      endif

                      if MPM
                              mov a,c! sui 3
                              mov l,a! mvi h,0! dad h
                              lxi d,xerr$list! dad d
                              mov e,m! inx h! mov d,m
                              xchg! jmp report$err
                      endif

                      func23:
                              ; Rename a file
1989 CDA516                   call reselectx
198C C33A10                   jmp rename

                      func24:
                              ; Return the login vector
198F 2A881CC3C3               lhld dlog! jmp sthl$ret

                      func25:
                              ; Return selected disk number
1995 3AC61CC3CB               lda seldsk! jmp sta$ret

                      func26:

                      if MPM
                              ; Save dma address in process descriptor
                              lhld info
```

```
               intrnlsetdma:
                       xchg
                       call rlr! lxi b,disksetdma! dad b
                       mov m,e! inx h! mov m,d
               endif

                       ; Set the subsequent dma address to info
199B EB22D81E          xchg! shld dmaad ; dmaad = info
199F C3520C             jmp setdata ; to data dma address

               func27:
                       ; Return the login vector address
19A2 CD8016            call curselect
19A5 2A9C1CC3C3        lhld alloca! jmp sthl$ret

               if MPM

               func28:
                       ; Write protect current disk
                       ; first check for open files on disk
                       mvi a,0ffh! sta set$ro$flag
                       lda seldsk! mov c,a! lxi h,0001h
                       call hlrotl! call intrnldiskreset
                       jmp set$ro
               else

0B91 =         func28:        equ     set$ro ; Write protect current disk

               endif

               func29:
                       ; Return r/o bit vector
19AB 2A861CC3C3        lhld rodsk! jmp sthl$ret

               func30:
                       ; Set file indicators
19B1 CD1510            call check$wild
19B4 CDA516            call reselectx
19B7 CD7110            call indicators
19BA C3160E            jmp copy$dirloc ; lret=dirloc

               func31:
                       ; Return address of disk parameter block
19BD CD8016            call curselect
19C0 2A981C            lhld dpbaddr
               sthl$ret:
19C3 227A07C9          shld aret! ret

               func32:
                       ; Set user code
19C7 3AC31CFEFF         lda linfo! cpi 0ffh! jnz setusrcode
                               ; Interrogate user code instead
19CF 3AE01EC3CB                lda usrcode! jmp sta$ret ; lret=usrcode
                       setusrcode:
19D5 E60F32E01E                ani 0fh! sta usrcode

```

```
               if MPM
                              push a
                              call rlr! lxi b,diskselect! dad b
                              pop b
                              mov a,m! ani 0f0h! ora b! mov m,a
               endif

19DA C9                       ret

               func33:
                      ; Random disk read operation
19DB CDAC16           call reselect
19DE CD8C0A           call check$fcb
19E1 C3CE15           jmp randiskread ; to perform the disk read

               func34:
                      ; Random disk write operation
19E4 CDAC16           call reselect
19E7 CD8C0A           call check$fcb
19EA C3D715           jmp randiskwrite ; to perform the disk write

               func35:
                      ; Return file size (0-262,144)
19ED CDAC16           call reselect
19F0 C32816           jmp getfilesize

1655 =         func36 equ setrandom ; Set random record

               func37:
                      ; Drive reset

               if MPM
                      call diskreset
               reset$37:
                      lhld info
               else
19F3 EB               xchg
               endif

               reset$37x:
19F4 7D2F5F7C2F       mov a,l! cma! mov e,a! mov a,h! cma
19F9 2A881CA457       lhld dlog! ana h! mov d,a! mov a,l! ana e
1A00 5F2A861CEB       mov e,a! lhld rodsk! xchg! shld dlog

               if MPM
                      push h! call hl$eq$hl$and$de
               else
1A08 7DA36F           mov a,l! ana e! mov l,a
1A0B 7CA267           mov a,h! ana d! mov h,a
               endif

1A0E 22861C           shld rodsk

               if MPM
                      pop h! xchg! lhld rlog! call hl$eq$hl$and$de! shld rlog
               endif
```

CP/M -PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3 - 135

```

                        ; Force select call in next curselect
1A11 3EFF32CB1C         mvi a,0ffh! sta curdsk! ret

                if MPM

                func38:
                        ; Access drive

                        lxi h,packed$dcnt! mvi a,0ffh
                        mov m,a! inx h! mov m,a! inx h! mov m,a
                        xra a! xchg! lxi b,16
                acc$drv0:
                        dad h! adc b! dcr c! jnz acc$drv0
                        ora a! rz
                        sta mult$cnt! dcr a! push a
                        call acc$drv02
                        pop a! jmp openx02 ; insufficient free lock list items
                acc$drv02:
                        call check$free! pop h ; Discard return addr, free space exists
                        call count$opens! pop b! add b! jc openx034
                        sub m! jnc openx034 ; openmax exceeded
                        lhld info! lda curdsk! push a! mvi a,16
                acc$drv1:
                        dcr a! dad h! jc acc$drv2
                acc$drv15:
                        ora a! jnz acc$drv1
                        pop a! sta curdsk! ret
                acc$drv2:
                        push a! push h! sta curdsk
                        call create$olist$item
                        pop h! pop a! jmp acc$drv15

                func39:
                        ; Free drive
                        lhld open$root! mov a,h! ora l! rz
                        xra a! sta incr$pdcnt! inr a! sta free$mode
                        lhld info! mov a,h! cmp l! jnz free$drv1
                        inr a! jnz free$drv1
                        sta free$mode! call free$files! jmp free$drv3
                free$drv1:
                        lda curdsk! push a! mvi a,16
                free$drv2:
                        dcr a! dad h! jc free$drv4
                free$drv25:
                        ora a! jnz free$drv2
                        pop a! sta curdsk
                free$drv3:
                        lda incr$pdcnt! ora a! rz
                        lda pdcnt! jmp chk$olist1
                free$drv4:
                        push a! push h! sta curdsk
                        call free$files
                        pop h! pop a! jmp free$drv25
                else

```

```
06CE =          func38          equ     func$ret
06CE =          func39 equ      func$ret

                endif

19E4 =          func40  equ     func34          ; Write random with zero fill

                if MPM

                func41 equ      func$ret        ; Test & write
                func42:                                 ; Record lock
                        mvi a,0ffh! sta lock$unlock! jmp lock
                func43:                                 ; Record unlock
                        xra a! sta lock$unlock! jmp lock

                else

06CE =          func42 equ      func$ret        ; Record lock
06CE =          func43 equ      func$ret        ; Record unlock

                endif

                func44:                                 ; Set multi-sector count
1A17 7BB7CA260F         mov a,e! ora a! jz lret$eq$ff
1A1C FE81D2260F         cpi 129! jnc lret$eq$ff
1A21 32E61E             sta mult$cnt

                if MPM
                        mov d,a
                        call rlr! lxi b,mult$cnt$off! dad b
                        mov m,d
                endif

1A24 C9                 ret

                func45:                                 ; Set bdos error mode

                if MPM
                        call rlr! lxi b,pname+4! dad b
                        call set$pflag
                        mov m,a! inx h
                        call set$pflag
                        mov m,a! ret

                set$pflag:
                        mov a,m! ani 7fh! inr e! rnz
                        ori 80h! ret
                else
1A25 7B32E71E           mov a,e! sta error$mode
                endif

1A29 C9                 ret

                func46:
                        ; Get free space
                        ; Perform temporary select of specified drive
```

```
1A2A CD7C16            call tmpselect
1A2D 2A9C1CEB          lhld alloca! xchg ; de = alloc vector addr
1A31 CD860D            call get$nalbs ; Get # alloc blocks
                       ; hl = # of allocation vector bytes
                       ; Count # of true bits in allocation vector
1A34 010000            lxi b,0 ; bc = true bit accumulator
1A37 1A         gsp1:  ldax d
1A38 B7CA441A   gsp2:  ora a! jz gsp4
1A3C 1FD23C1A   gsp3:  rar! jnc gsp3
1A40 03C3381A          inx b! jmp gsp2
1A44 132B       gsp4:  inx d! dcx h
1A46 7DB4C2371A        mov a,l! ora h! jnz gsp1
                       ; hl = 0 when allocation vector processed
                       ; Compute maxall + 1 - bc
1A4B 2AAA1C23          lhld maxall! inx h
1A4F 7D916F            mov a,l! sub c! mov l,a
1A52 7C9867            mov a,h! sbb b! mov h,a
                       ; hl = # of available blocks on drive
1A55 3AA71C4FAF        lda blkshf! mov c,a! xra a
1A5A CDC207            call shl3bv
                       ; ahl = # of available sectors on drive
                       ; Store ahl in beginning of current dma
1A5D EB2A8A1C73        xchg! lhld xdmaad! mov m,e! inx h
1A63 722377C9          mov m,d! inx h! mov m,a! ret

                if MPM

                func47 equ     func$ret

                else

                func47:                             ; Chain to program
1A67 21B31E7EF6        lxi h,ccp$flgs! mov a,m! ori 80h! mov m,a
1A6E 1CC2C802          inr e! jnz rebootx1
1A72 7EF64077          mov a,m! ori 40h! mov m,a
1A76 C3C802            jmp rebootx1
                endif

                func48:                                     ; Flush buffers
1A79 CD3617            call check$all$media
1A7C CD48FF            call flushf
1A7F CD7108            call diocomp
                flush0:                                     ; Function 98 entry point
1A82 2A881C3E10        lhld dlog! mvi a,16
                flush1:
1A87 3D29D2AF1A        dcr a! dad h! jnc flush5
1A8C F5E55FCD7C        push a! push h! mov e,a! call tmpselect ; seldsk = e
1A92 3ADF1EFE30        lda fx! cpi 48! jz flush3
                       ; Function 98 - reset allocation
                       ; Copy 2nd ALV over 1st ALV
1A9A CD130DC3AA        call copy$alv! jmp flush35
                flush3:
1AA0 CDBA1A            call flushx
                       ; if e = 0ffh then discard buffers after possible flush
1AA3 3AC31C3CC2        lda linfo! inr a! jnz flush4
                flush35:
```

```
1AAA CD100C           call discard$data
               flush4:
1AAD E1F1             pop h! pop a
               flush5:
1AAF B7C2871A         ora a! jnz flush1
1AB3 C9               ret

               flush:
1AB4 CD48FF           call flushf
1AB7 CD7108           call diocomp
               flushx:
1ABA 3AB51CB7C8       lda phymsk! ora a! rz
1ABF 3E04C3021D       mvi a,4! jmp deblock$dta

               if MPM

               func49 equ     func$ret

               else

               func49:        ; Get/Set system control block

1AC4 EB7EFE63D0       xchg! mov a,m! cpi 99! rnc
1AC9 EB219C1E85       xchg! lxi h,scb! add l! mov l,a
1ACF EB237EFEFE       xchg! inx h! mov a,m! cpi 0feh! jnc func49$set
1AD7 EB5E2356EB       xchg! mov e,m! inx h! mov d,m! xchg
1ADC C3C319           jmp sthl$ret
               func49$set:
1ADF 47237E1204       mov b,a! inx h! mov a,m! stax d! inr b! rz
1AE5 23137E12C9       inx h! inx d! mov a,m! stax d! ret
               endif

               if MPM

               func50 equ     func$ret

               else

               func50:                                  ; Direct bios call
                      ; de -> function (1 byte)
                      ;       a  value (1 byte)
                      ;       bc value (2 bytes)
                      ;       de value (2 bytes)
                      ;       hl value (2 bytes)

1AEA 210F1BE5         lxi h,func50$ret! push h
1AEE EB               xchg

               if BANKED
                      mov a,m! cpi 27! rz
                      cpi 12! jnz dir$bios1
                      lxi d,dir$bios3! push d
               dir$bios1:
                      cpi 9! jnz dir$bios2
                      lxi d,dirbios4! push d
               dir$bios2:
```

```

                        endif

1AEF E52323                     push h! inx h! inx h
1AF2 4E234623                   mov c,m! inx h! mov b,m! inx h
1AF6 5E235623                   mov e,m! inx h! mov d,m! inx h
1AFA 7E23666F                   mov a,m! inx h! mov h,m! mov l,a
1AFE E37EE56F87                 xthl! mov a,m! push h! mov l,a! add a! add l

1B04 2100FF                     lxi h,bios

1B07 856FE3                     add l! mov l,a! xthl
1B0A 237EE1E3C9                 inx h! mov a,m! pop h! xthl! ret

                        if BANKED

                        dir$bios3:
                                mvi a,1! jmp setbnkf

                        dir$bios4:
                                mov a,l! ora h! rz
                                xchg! lxi h,10! dad d! mvi m,0 ; Zero login sequence #
                                lhld common$base! call subdh! xchg! rnc
                                ; Copy DPH to common memory
                                xchg! lhld info! inx h! push h! lxi b,25
                                call movef! pop h! ret
                        endif

                        func50$ret:

                        if BANKED
                                shld aret! mov b,a
                                lhld info! mov a,m
                                cpi 9! rz
                                cpi 16! rz
                                cpi 20! rz
                                cpi 22! rz
                                mov a,b! jmp sta$ret
                        else
1B0F EB2ACB02F9                 xchg! lhld entsp! sphl! xchg! ret
                        endif
                        endif

1A82 =                  func98          equ     flush0                  ; Reset Allocation

                        func99:                                         ; Truncate file
1B16 CDA516                     call reselectx
1B19 CD1510                     call check$wild

                        if BANKED
                                call chk$password! cnz chk$pw$error
                        endif

1B1C 0EFFCD0015                 mvi c,true! call rseek! jnz lret$eq$ff
                                ; compute dir$fcb size
1B24 CDFE0A110F                 call getdptra! lxi d,reccnt
```

```
1B2A CDE015          call compute$rr ; cba = fcb size
                     ; Is random rec # >= dir$fcb size
1B2D CDF509CD13      call get$rra! call compare$rr
1B33 DA260F          jc lret$eq$ff ; yes ( > )
1B36 B2CA260F        ora d! jz lret$eq$ff  ; yes ( = )
                     ; Perform truncate
1B3A CDE60A          call check$rodir ; may be r/o file
1B3D CD3E0C          call wrdir ; verify BIOS can write to disk
1B40 CDE817          call update$stamp ; Set update stamp
1B43 CD8B0E          call search$extnum
              trunc1:
1B46 CA160E          jz copy$dirloc
                     ; is dirfcb < fcb?
1B49 CDB91BDA64      call compare$mod$ext! jc trunc2 ; yes
                     ; remove dirfcb blocks from allocation vector
1B4F F50E00CD2D      push a! mvi c,0! call scandm$ab! pop a
                     ; is dirfcb = fcb?
1B56 CA6A1B          jz trunc3 ; yes
                     ; delete dirfcb
1B59 CDFE0A36E5      call getdptra! mvi m,empty! call fix$hash
              trunc15:
1B61 CD3E0C          call wrdir
              trunc2:
1B64 CD990E          call searchn
1B67 C3461B          jmp trunc1
              trunc3:
1B6A CD160ACD62      call getfcb! call dm$position
1B70 CDD31B          call zero$dm
                     ; fcb(extnum) = dir$ext after blocks removed
1B73 CD2D0EBE77      call get$dir$ext! cmp m! mov m,a! push a
                     ; fcb(rc) = fcb(cr) + 1
1B79 CD0D0A7E3C      call getfcba! mov a,m! inr a! stax d
                     ; rc = 0 or 128 if dir$ext < fcb(extnum)
1B7F F1EBC4DC10      pop a! xchg! cnz set$rc3
                     ; rc = 0 if no blocks remain in fcb
1B84 3AC41CB7CC      lda dminx! ora a! cz set$rc3
1B8B 010B00CDAF      lxi b,11! call get$fcb$adds! xchg
                     ; reset archive (t3') attribute bit
1B92 7EE67F7723      mov a,m! ani 7fh! mov m,a! inx h! inx d
                     ; dirfcb(extnum) = fcb(extnum)
1B98 1A77            ldax d! mov m,a
                     ; advance to .fcb(reccnt) & .dirfcb(reccnt)
1B9A 2336002323      inx h! mvi m,0! inx h! inx h
1B9F 131313          inx d! inx d! inx d
                     ; dirfcb_rc+dskmap = fcb_rc+dskmap
1BA2 0E11CDF507      mvi c,17! call move
                     ; restore non-erased blkidxs in allocation vector
1BA7 0E01CD2D0D      mvi c,1! call scandm$ab
1BAC C3611B          jmp trunc15

              get$fcb$adds:
1BAF CDFE0A09EB      call getdptra! dad b! xchg
1BB4 2ADB1E09C9      lhld info! dad b! ret

              compare$mod$ext:
1BB9 010E00CDAF      lxi b,modnum! call get$fcb$adds
```

```
1BBF 7EE63F47           mov a,m! ani 3fh! mov b,a
                        ; compare dirfcb(modnum) to fcb(modnum)
1BC3 1AB8C0             ldax d! cmp b! rnz ; dirfcb(modnum) "= fcb(modnum)
1BC6 2B2B1B1B           dcx h! dcx h! dcx d! dcx d
                        ; compare dirfcb(extnum) to fcb(extnum)
1BCA 1A4ECD1C0E         ldax d! mov c,m! call compext! rz ; dirfcb(extnum) = fcb(extnum)
1BD0 1ABEC9             ldax d! cmp m! ret

                  zero$dm:
1BD3 3C21C51C34         inr a! lxi h,single! inr m! jz zero$dm1
1BDB 87                 add a
                  zero$dm1:
1BDC 35                 dcr m
1BDD CD82094F06         call getdma! mov c,a! mvi b,0! dad b
1BE4 3E10               mvi a,16
                  zero$dm2:
1BE6 B9C8               cmp c! rz
1BE8 70230CC3E6         mov m,b! inx h! inr c! jmp zero$dm2

                  if BANKED

                  func100:                          ; Set directory label
                        ; de -> .fcb
                        ;       drive location
                        ;       name & type fields user's discretion
                        ;       extent field definition
                        ;       bit 1 (80h): enable passwords on drive
                        ;       bit 2 (40h): enable file access
                        ;       bit 3 (20h): enable file update stamping
                        ;       bit 4 (10h): enable file create stamping
                        ;       bit 8 (01h): assign new password to dir lbl
                        call reselectx
                        lhld info! mvi m,21h! mvi c,1
                        call search! jnz sd10
                        call getexta! mov a,m! ani 0111$0000b! jnz lret$eq$ff
                  sd10:
                        ; Does dir lbl exist on drive?
                        lhld info! mvi m,20h! mvi c,1
                        call set$xdcnt! call search! jnz sd11
                        ; no - make one
                        mvi a,0ffh! sta make$xfcb
                        call make! rz ; no dir space
                        call init$xfcb
                        lxi b,24! call stamp5! call stamp1
                  sd11:
                        ; Update date & time stamp
                        lxi b,28! call stamp5! call stamp2
                        ; Verify password - new dir lbl falls through
                        call chk$xfcb$password! jnz pw$error
                        lxi b,0! call init$xfcb0
                        ; Set dir lbl dta in extent field
                        ldax d! ori 1h! mov m,a
                        ; Low bit of dir lbl data set to indicate dir lbl exists
                        ; Update drive's dir lbl vector element
                        push h! lhld drvlbla! mov m,a! pop h
                  sd12:
```

```
                                ; Assign new password to dir lbl or xfcb?
                                ldax d! ani 1! jz sdl3
                                ; yes - new password field is in 2nd 8 bytes of dma
                                lxi d,8! call adjust$dmaad
                                call set$pw! mov m,b
                                lxi d,-8! call adjust$dmaad
                        sdl3:
                                call fix$hash
                                jmp seek$copy
                        else

 0F26 =                 func100         equ     lret$eq$ff
 0F26 =                 func103 equ     lret$eq$ff

                        endif

                        func101:
                                ; Return directory label data
                                ; Perform temporary select of specified drive
 1BEE CD7C16                    call tmpselect
 1BF1 CD6D17C3CB                call get$dir$mode! jmp sta$ret

                        func102:
                                ; Read file xfcb
 1BF7 CDA516                    call reselectx
 1BFA CD1510                    call check$wild
 1BFD CD5A0A                    call zero$ext$mod
 1C00 CD860EC8                  call search$namlen! rz
 1C04 CD82090108                call getdma! lxi b,8! call zero
 1C0D E50E00CDA7                push h! mvi c,0! call get$dtba! ora a! jnz rxfcb2
 1C17 D1EB0E08                  pop d! xchg! mvi c,8

                        if BANKED
                                call move! ldax d! jmp rxfcb3
                        else
 1C1B C3F507                    jmp move
                        endif

                        rxfcb2:
 1C1E E1010800                  pop h! lxi b,8

                        if BANKED
                                call zero! call get$xfcb! rz
                                mov a,m
                        rxfcb3:
                                call getexta! mov m,a! ret
                        else
 1C22 C3620A                    jmp zero
                        endif

                        if BANKED

                        func103:
                                ; Write or update file xfcb
                                call reselectx
                                ; Are passwords enabled in directory label?
```

```
          call get$dir$mode! ral! jnc lret$eq$ff ; no
          call check$wild
          ; Save .fcb(ext) & ext
          call getexta! mov b,m! push h! push b
          ; Set extent & mod to zero
          call zero$ext$mod
          ; Does file's 1st fcb exist in directory?
          call search$namlen
          ; Restore extent
          pop b! pop h! mov m,b! rz ; no
          call set$xdcnt
          ; Does sfcb exist?
          call get$dtba$8! ora a! jz wxfcb5 ; yes
          ; No - Does xfcb exist?
          call get$xfcb! jnz wxfcb1 ; yes
wxfcb0:
          ; no - does file exist in directory?
          mvi a,0ffh! sta make$xfcb
          call search$extnum! rz
          ; yes - attempt to make xfcb for file
          call make! rz ; no dir space
          ; Initialize xfcb
          call init$xfcb
wxfcb1:
          ; Verify password - new xfcb falls through
          call chk$xfcb$password! jnz pw$error
          ; Set xfcb options data
          push h! call getexta! pop d! xchg
          mov a,m! ora a! jnz wxfcb2
          ldax d! ani 1! jnz wxfcb2
          call sdl3! jmp wxfcb4
wxfcb2:
          ldax d! ani 0e0h! jnz wxfcb3
          mvi a,80h
wxfcb3:
          mov m,a! call sdl2
wxfcb4:
          call get$xfcb1! dcr a! sta pw$mode
          call zero$ext$mod
          call search$namlen! rz
          call get$dtba$8! ora a! rnz
          lda pw$mode! mov m,a! jmp seek$copy
wxfcb5:
          ; Take sfcb's password mode over xfcb's mode
          mov a,m! push a
          call get$xfcb
          ; does xfcb exist?
          pop b! jz wxfcb0 ; no
          ; Set xfcb's password mode to sfcb's mode
          mov m,b! jmp wxfcb1

endif

func104:                        ; Set current date and time

if MPM
```

```
                                call get$stamp$add
                        else
1C25 21F41E                     lxi h,stamp
                        endif
1C28 CD421C                     call copy$stamp
1C2B 36000EFFC3                 mvi m,0! mvi c,0ffh! jmp timef

                        func105:                        ; Get current date and time



                        if MPM
                                call get$stamp$add
                        else
1C32 0E00CD4EFF                 mvi c,0! call timef
1C37 21F41E                     lxi h,stamp
                        endif

1C3A EB                         xchg
1C3B CD421C                     call copy$stamp
1C3E 1AC3CB06                   ldax d! jmp sta$ret

                        copy$stamp:
1C42 0E04C3F507                 mvi c,4! jmp move ; ret

                        if MPM

                        get$stamp$add:
                                call rlradr! lxi b,-5! dad b
                                ret
                        endif

                        if BANKED

                        func106:                        ; Set default password

                        if MPM
                                call get$df$pwa! inr a! rz
                                lxi b,7! dad b
                        else
                                lxi h,df$password+7
                        endif
                                xchg! lxi b,8! push h
                                jmp set$pw0
                        else

06CE =                  func106         equ     func$ret

                        endif

                        func107:                        ; Return serial number

                        if MPM
                                lhld sysdat! mvi l,181
                        else
1C47 210000                     lxi h,serial
```

```
                      endif

1C4A EB0E06C3F5                xchg! mvi c,6! jmp move

                      func108:                        ; Get/Set program return code

                               ; Is de = 0ffffh?
1C50 7AA33C                    mov a,d! ana e! inr a
1C53 2AAC1ECAC3                lhld clp$errcde! jz sthl$ret ; yes - return return code
1C59 EB22AC1EC9                xchg! shld clp$errcde! ret ; no - set return code

                      goback0:
1C5E 21FFFF227A                lxi h,0ffffh! shld aret
                      goback:
                               ; Arrive here at end of processing to return to user
1C64 3ADD1EB7CA                lda resel! ora a! jz retmon

                      if MPM
                                           lda comp$fcb$cks! ora a! cnz set$chksum$fcb
                      endif

1C6B 2ADB1E3AE3                            lhld info! lda fcbdsk! mov m,a ; fcb(0)=fcbdsk
                      if BANKED

                                           ; fcb(7) = fcb(7) | xfcb$read$only
                                           lxi d,7! dad d! lda xfcb$read$only! ora m! mov m,a

                      endif
                      if MPM
                                           ; if high$ext = 60h then fcb(8) = fcb(8) | 80h
                                           ;                      else fcb(ext) = fcb(ext) | high$ext

                                           call getexta! lda high$ext! cpi 60h! jnz goback2
                                           lxi d,-4! dad d! mvi a,80h
                               goback2:
                                           ora m! mov m,a
                      else
                                           ; fcb(8) = fcb(8) | high$ext
                      if BANKED
                                           inx h
                      else
1C72 11080019                              lxi d,8! dad d
                      endif
1C76 3AD71CB677                            lda high$ext! ora m! mov m,a
                      endif

                      ;        return from the disk monitor

                      retmon:
1C7B 2ACB02F9                  lhld entsp! sphl
1C7F 2A7A077D44                lhld aret! mov a,l! mov b,h! ret
                      ;
                      ;        data areas
                      ;
1C85 E5               efcb:  db       empty    ; 0e5=available dir entry
1C86 0000             rodsk: dw       0        ; read only disk vector
```

```
1C88 0000       dlog:  dw      0       ; logged-in disks

                if MPM

                rlog:  dw      0       ; removeable logged-in disks
                tlog:  dw      0       ; removeable disk test login vector
                ntlog: dw      0       ; new tlog vector
                rem$drv: ds    byte    ; curdsk removable drive switch
                                       ; 0 = permanent drive, 1 = removable drive
                endif

                if not BANKED

1C8A =          xdmaad equ     $
1C8A            curdma ds      word    ; current dma address

                endif

                if not MPM

1C8C            buffa: ds      word    ; pointer to directory dma address

                endif

                ;
                ;       curtrka - alloca are set upon disk select
                ;       (data must be adjacent, do not insert variables)
                ;       (address of translate vector, not used)
1C8E            cdrmaxa:ds     word    ; pointer to cur dir max value (2 bytes)
1C90            curtrka:ds     word    ; current track address (2)
1C92            curreca:ds     word    ; current record address (3)
1C94            drvlbla:ds     word    ; current drive label byte address (1)
1C96            lsn$add:ds     word    ; login sequence # address (1)
                                       ; +1 -> bios media change flag (1)
1C98            dpbaddr:ds     word    ; current disk parameter block address
1C9A            checka:        ds      word    ; current checksum vector address
1C9C            alloca:        ds      word    ; current allocation vector address
1C9E            dirbcba:ds     word    ; dir bcb list head
1CA0            dtabcba:ds     word    ; data bcb list head
                hash$tbla:
1CA2                   ds      word    ; directory hash table address
1CA4                   ds      byte    ; directory hash table bank

000D =          addlist        equ     $-dpbaddr       ; address list size

                ;
                ;              buffer control block format
                ;
                ; bcb format : drv(1) || rec(3) || pend(1) || sequence(1) ||
                ;              0         1         4           5
                ;
                ;              track(2) || sector(2) || buffer$add(2) ||
                ;              6           8            10
                ;
                ;              bank(1) || link(2)
                ;              12         13
                ;

                ;       sectpt - offset obtained from disk parm block at dpbaddr
                ;       (data must be adjacent, do not insert variables)
 1CA5           sectpt:         ds      word    ; sectors per track
 1CA7           blkshf:         ds      byte    ; block shift factor
 1CA8           blkmsk:         ds      byte    ; block mask
 1CA9           extmsk:         ds      byte    ; extent mask
 1CAA           maxall:         ds      word    ; maximum allocation number
 1CAC           dirmax:         ds      word    ; largest directory number
 1CAE           dirblk:         ds      word    ; reserved allocation bits for directory
 1CB0           chksiz:         ds      word    ; size of checksum vector
 1CB2           offset:         ds      word    ; offset tracks at beginning
 1CB4           physhf:         ds      byte    ; physical record shift
 1CB5           phymsk:         ds      byte    ; physical record mask
 0011 =         dpblist         equ     $-sectpt        ; size of area
                ;
                ;       local variables
                ;
 1CB6           drec            ds      word    ; directory record number
 1CB8           blk$off:        ds      byte    ; record offset within block
 1CB9           last$off:       ds      byte    ; last offset within new block
 1CBA           last$drive:     ds      byte    ; drive of last new block
 1CBB           last$block:     ds      word    ; last new block

                ; The following two variables are initialized as a pair on entry

 1CBD           dir$cnt:        ds      byte    ; direct i/o count
 1CBE           mult$num:       ds      byte    ; multi-sector number

 1CBF           tranv: ds       word    ; address of translate vector
                lock$unlock:
                make$flag:
 1CC1           rmf:   ds       byte    ; read mode flag for open$reel
                incr$pdcnt:
 1CC2           dirloc:         ds      byte    ; directory flag in rename, etc.
                free$mode:
 1CC3           linfo: ds       byte    ; low(info)
 1CC4           dminx: ds       byte    ; local for diskwrite

                if MPM

                searchl:ds      byte    ; search length

                endif
                if BANKED

                searcha:ds      word    ; search address

                endif

                if BANKED

                save$xfcb:
                        ds      byte    ; search xfcb save flag

```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SFR. #

```
                endif

1CC5            single:         ds      byte    ; set true if single byte allocation map

                if MPM

                seldsk: ds      byte    ; currently selected disk

                endif

1CC6            seldsk:         ds      byte    ; disk on entry to bdos
1CC7            rcount:         ds      byte    ; record count in current fcb
1CC8            extval:         ds      byte    ; extent number and extmsk
                save$mod:
1CC9                    ds      byte    ; open$reel module save field

1CCA            vrecord:ds      byte    ; current virtual record

                if not MPM

1CCB FF         curdsk: db      0ffh    ; current disk

                endif

1CCC FF         adrive: db      0ffh    ; current blocking/deblocking disk
1CCD            arecord:ds      word    ; current actual record
1CCF                    ds      byte

1CD0            save$ranr:      ds      3       ; random record save area
1CD3            arecord1:       ds      word    ; current actual block# * blkmsk
1CD5            attributes:     ds      byte    ; make attribute hold area
1CD6            readf$sw:       ds      byte    ; BIOS read/write switch

                ;******** following variable order critical *****************

                if MPM

                mult$cnt:       ds      byte    ; multi-sector count
                pdcnt:          ds      byte    ; process descriptor count

                endif

1CD7            high$ext:       ds      byte    ; fcb high ext bits

                if BANKED

                xfcb$read$only:         ds      byte    ; xfcb read only flag

                endif
                if MPM

                curdsk:                 db      0ffh                    ;current disk
                packed$dcnt:    ds      3                       ;
                pdaddr:                 ds      word                    ;
                ;**************************************************************
                cur$pos:        ds      word                    ;
                prv$pos:        ds      word                    ;
                sdcnt:          ds      word                    ;
                sdblk:          ds      word                    ;
                sdcnt0:                 ds      word                    ;
                sdblk0:                 ds      word                    ;
                dont$close:     ds      byte                    ;
                open$cnt:                       ; mp/m temp variable for open
                lock$cnt:       ds      word    ; mp/m temp variable for lock
                file$id:        ds      word    ; mp/m temp variable for lock
                deleted$files:  ds      byte
                lock$shell:     ds      byte
                lock$sp:        ds      word
                set$ro$flag:    ds      byte
                check$disk:     ds      byte
                flushed:        ds      byte
                fcb$cks$valid:  ds      byte
                ;                               mp/m variables  *

                endif

                ;       local variables for directory access
1CD8            dptr:   ds      byte    ; directory pointer 0,1,2,3

1CD9            save$hash:      ds      4       ; hash code save area

                if BANKED

                copy$cr$init:   ds      byte    ; copy$cr$only initialization value

                else

1CDD            hashmx:         ds      word    ; cdrmax or dirmax
1CDF            xdcnt:  ds      word    ; empty directory dcnt

                endif

                if MPM

                xdcnt:  ds      word    ; empty directory dcnt
                xdblk:  ds      word    ; empty directory block
                dcnt:   ds      word    ; directory counter 0,1,...,dirmax
                dblk:   ds      word    ; directory block index

                endif

1CE1            search$user0:   ds      byte    ; search user 0 for file (open)

1CE2            user0$pass:     ds      byte    ; search user 0 pass flag

1CE3            fcbdsk:                 ds      byte    ; disk named in fcb

                if MPM

                make$xfcb:      ds      1
                find$xfcb:      ds      1

```

```
               endif

1CE4 0F10111316log$fxs:db     15,16,17,19,22,23,30,35,99,100,102,103,0
1CF1 1415212228rw$fxs: db     20,21,33,34,40,41,0
1CF8 101200     sc$fxs: db     16,18,0

               if MPM

               comp$fcb$cks:  ds      byte    ; compute fcb checksum flag

               endif
               if BANKED

               pw$fcb:                ds      12      ;1 !
                              db      0       ;2 !
               pw$mode:       db      0       ;3 !- Order critical
                              db      0       ;4 !
                              db      0       ;5 !

               df$password:   ds      8

               if MPM
                              ds      120
               endif
               endif

1CFB           phy$off:       ds      byte
1CFC           curbcba:       ds      word

               if BANKED

               lastbcba:      ds      word
               rootbcba:      ds      word
               emptybcba:     ds      word
               seqbcba:       ds      word
               buffer$bank:   ds      byte

               endif

1CFE           track:         ds      word
1D00           sector:                ds      word

               ;      *************************
               ;      Blocking/Deblocking Module
               ;      *************************

               deblock$dta:
1D02 2AA01C           lhld dtabcba

               if BANKED
                      cpi 4! jnz deblock
               deblock$flush:
                      ; de = addr of 1st bcb
                      mov e,m! inx h! mov d,m
                      ; Search for dirty bcb with lowest track #
                      lxi h,0ffffh! shld track! xchg
```

```
                    deblock$flush1:
                            ; Does current drive own bcb?
                            lda adrive! cmp m! jnz deblock$flush2 ;no
                            ; Is bcb's buffer pending?
                            xchg! lxi h,4! dad d! mov a,m
                            xchg! inr a! jnz deblock$flush2 ; no
                            ; Is bcb(6) < track?
                            push h! inx d! inx d! xchg
                            mov e,m! inx h! mov d,m
                            ; Subdh computes hl = de - hl
                            lhld track! call subdh! pop h! jnc deblock$flush2 ; no
                            ; yes - track = bcb(6) , sector = addr(bcb)
                            xchg! shld track! xchg! shld sector
                    deblock$flush2:
                            ; Is this the last bcb?
                            call get$next$bcba! jnz deblock$flush1 ; no - hl = addr of next bcb
                            ; Does track = ffff?
                            lxi h,track! call test$ffff! rz ; yes - no bcb to flush
                            ; Flush bcb located by sector
                            lhld sector! xra a! mvi a,4! call deblock
                            lhld dtabcba! jmp deblock$flush ; Repeat until no bcb's to flush
                    endif

                    deblock:

                            ; BDOS Blocking/Deblocking routine
                            ; a = 1 -> read command
                            ; a = 2 -> write command
                            ; a = 3 -> locate command
                            ; a = 4 -> flush command
                            ; a = 5 -> directory update

1D05 F5                     push a ; Save z flag and deblock fx

                            ; phy$off = low(arecord) & phymsk
                            ; low(arecord) = low(arecord) & ~phymsk
1D06 CDCA1D                 call deblock8
1D09 3ACD1C5FA0             lda arecord! mov e,a! ana b! sta phy$off
1D11 7BA132CD1C             mov a,e! ana c! sta arecord

                    if BANKED
                            pop a! push a! cnz get$bcba
                    endif

1D16 22FC1CCD2B             shld curbcba! call getbuffa! shld curdma
                            ; hl = curbcba, de = .adrive, c = 4
1D1F CDD11D                 call deblock9
                            ; Is BCB discarded?
1D22 7E3CCA6A1D             mov a,m! inr a! jz deblock2 ; yes
                            ; Is command flush?
1D27 F1F5FE04D2             pop a! push a! cpi 4! jnc deblock1 ; yes
                            ; Is referenced physical record already in buffer?
1D2E CDEB07CA96             call compare! jz deblock45 ; yes
1D34 AF                     xra a
                    deblock1:
                            ; Does buffer contain an updated record?
```

```
1D35 CDDA1D             call deblock10
1D38 FE05CA421D         cpi 5! jz deblock15
1D3D 7EB7CA6A1D         mov a,m! ora a! jz deblock2 ; no
                  deblock15:
                        ; Reset record pending flag
1D42 3600               mvi m,0
                        ; Save arecord
1D44 2ACD1CE53A         lhld arecord! push h! lda arecord+2! push a
                        ; Flush physical record buffer
1D4C CDD11D             call deblock9
1D4F EBCDF507           xchg! call move
                        ; Select drive to be flushed
1D53 21CB1C3ACC         lxi h,curdsk! lda adrive! cmp m! cnz disk$select1
                        ; Write record if drive logged-in
1D5D 3E01CCE21D         mvi a,1! cz deblock$io
                        ; Restore arecord
1D62 C1D1CDC108         pop b! pop d! call set$arecord
                        ; Restore selected drive
1D67 CD8016             call curselect
                  deblock2:
                        ; Is deblock command flush ¦ dir write?
1D6A F1FE04D0           pop a! cpi 4! rnc ; yes - return
                        ; Is deblock command write?
1D6E F5FE02C27E         push a! cpi 2! jnz deblock25 ; no
                        ; Is blk$off < last$off
1D74 21B91C3AB8         lxi h,last$off! lda blk$off! cmp m! jnc deblock3 ; no
                  deblock25:
                        ; Discard BCB on read operations in case
                        ; I/O error occurs
1D7E 2AFC1C36FF         lhld curbcba! mvi m,0ffh
                        ; Read physical record buffer
1D83 3E02C38B1D         mvi a,2! jmp deblock35
                  deblock3:
                        ; last$off = blk$off + 1
1D88 3C77               inr a! mov m,a
                        ; Place track & sector in bcb
1D8A AF                 xra a
                  deblock35:
1D8B CDE21D             call deblock$io
                  deblock4:
1D8E CDD11D             call deblock9 ; phypfx = adrive ¦¦ arecord
1D91 CDF5073600         call move! mvi m,0 ; zero pending flag

                  if BANKED
                        ; Zero logical record sequence
                        inx h! call set$bcb$seq
                  endif

                  deblock45:
                        ; recadd = phybuffa + phy$off*80h
1D96 3AFB1C3C11         lda phy$off! inr a! lxi d,80h! lxi h,0ff80h
                  deblock5:
1DA0 193DC2A01D         dad d! dcr a! jnz deblock5
1DA5 EB2A8A1C19         xchg! lhld curdma! dad d
                        ; If deblock command = locate then buffa = recadd; return
1DAA F1FE03C2B4         pop a! cpi 3! jnz deblock6
```

```
1DB0 228C1CC9          shld buffa! ret
               deblock6:
1DB4 EB2AD81E01        xchg! lhld dmaad! lxi b,80h
                       ; If deblock command = read
1DBB FE01              cpi 1

               if BANKED
                       jnz deblock7
                       ; then move to tpa
                       lda common$base+1! dcr a! cmp d! jc move$tpa
                       lda buffer$bank! mov c,a! mvi b,1! call deblock12
                       lxi b,80h! jmp move$tpa
               deblock7:

               else
1DBD CA4BFF            jz move$tpa ; then move to dma
               endif

                       ; else move from dma
1DC0 EB                xchg

               if BANKED
                       lda common$base+1! dcr a! cmp h! jc deblock75
                       lda buffer$bank! mov b,a! mvi c,1! call deblock12
                       lxi b,80h
               deblock75:

               endif

1DC1 CD4BFF            call move$tpa
                       ; Set physical record pending flag for write command
1DC4 CDDA1D36FF        call deblock10! mvi m,0ffh
1DC9 C9                ret

               deblock8:
1DCA 3AB51C472F        lda phymsk! mov b,a! cma! mov c,a! ret

               deblock9:
1DD1 2AFC1C11CC        lhld curbcba! lxi d,adrive! mvi c,4! ret

               deblock10:
1DDA 110400            lxi d,4
               deblock11:
1DDD 2AFC1C19C9        lhld curbcba! dad d! ret

               if BANKED

               deblock12:
                       push h! push d! call xmovef
                       pop d! pop h! ret
               endif

               deblock$io:
                       ; a = 0 -> seek only
                       ; a = 1 -> write
                       ; a = 2 -> read
```

```
1DE2 F5CDCA08           push a! call seek

                if BANKED
                        lda buffer$bank! call setbnkf
                endif

1DE6 0E01               mvi c,1
1DE8 F13D               pop a! dcr a
1DEA CA6A08             jz wrbuff
1DED F45F08             cp rdbuff
                        ; Move track & sector to bcb
1DF0 CDDA1D2323         call deblock10! inx h! inx h
1DF5 11FE1C0E04         lxi d,track! mvi c,4! jmp move

                if BANKED

                get$bcba:
                        shld rootbcba
                        lxi d,-13! dad d! shld lastbcba
                        call get$next$bcba! push h
                        ; Is there only 1 bcb in list?
                        call get$next$bcba! pop h! rz ; yes - return
                        xchg! lxi h,0! shld emptybcba! shld seqbcba
                        xchg
                get$bcb1:
                        ; Does bcb contain requested record?
                        shld curbcba! call deblock9! call compare! jz get$bcb4 ; yes
                        ; Is bcb discarded?
                        lhld curbcba! mov a,m! inr a! jnz get$bcb11 ; no
                        xchg! lhld lastbcba! shld emptybcba! jmp get$bcb14
                get$bcb11:
                        ; Does bcb contain record from current disk?
                        lda adrive! cmp m! jnz get$bcb15 ; no
                        xchg! lxi h,5! dad d! lda phy$msk
                        ; Is phy$msk = 0?
                        ora a! jz get$bcb14 ; yes
                        ; Does bcb(5) [bcb sequence] = phymsk?
                        cmp m! jnz get$bcb14 ; no
                        lhld seqbcba! mov a,l! ora h! jnz get$bcb14
                        lhld lastbcba! shld seqbcba
                get$bcb14:
                        xchg
                get$bcb15:
                        ; Advance to next bcb - list exhausted?
                        push h! call get$next$bcba! pop d! jz get$bcb2 ; yes
                        xchg! shld lastbcba! xchg! jmp get$bcb1
                get$bcb2:
                        ; Matching bcb not found
                        ; Was a sequentially accessed bcb encountered?
                        lhld seqbcba! mov a,l! ora h! jnz get$bcb25 ; yes
                        ; Was a discarded bcb encountered?
                        lhld emptybcba! mov a,l! ora h! jz get$bcb3 ; no
                get$bcb25:
                        shld lastbcba
                get$bcb3:
                        ; Insert selected bcb at head of list
```

CP/M - PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
                        lhld lastbcba! call get$next$bcba
                        shld curbcba! call get$next$bcba
                        xchg! call last$bcb$links$de
                        lhld rootbcba! mov e,m! inx h! mov d,m
                        lhld curbcba! lxi b,13! dad b
                        mov m,e! inx h! mov m,d
                        lhld curbcba! xchg! lhld rootbcba
                        mov m,e! inx h! mov m,d! xchg! ret
                get$bcb4:
                        ; BCB matched arecord
                        lhld curbcba! lxi d,5! dad d
                        ; Does bcb(5) = phy$off?
                        lda phy$off! cmp m! jz get$bcb5 ; yes
                        ; Does bcb(5) + 1 = phy$off?
                        inr m! cmp m! jz get$bcb5 ; yes
                        call set$bcb$seq
                get$bcb5:
                        ; Is bcb at head of list?
                        lhld curbcba! xchg! lhld rootbcba
                        mov a,m! inx h! mov l,m! mov h,a
                        call subdh! ora l! xchg! rz ; yes
                        jmp get$bcb3 ; no - insert bcb at head of list

                last$bcb$links$de:
                        lhld lastbcba! lxi b,13! dad b
                        mov m,e! inx h! mov m,d! ret

                get$next$bcba:
                        lxi b,13! dad b! mov e,m! inx h! mov d,m
                        xchg! mov a,h! ora l! ret

                set$bcb$seq:
                        lda phy$off! mov m,a! ora a! rz
                        lda phy$msk! inr a! mov m,a! ret

                endif

                if not MPM
                if not BANKED

1DFD                            ds      112
                last:
1E90                            org     base + (((last-base)+255) and 0ff00h) - 112

1E90 0000       olog:           dw      0
1E92 0000       rlog:           dw      0

1E94 00000000   patch$flgs:     dw      0,0
1E98 0600                       dw      base+6
1E9A AFC9                       xra a! ret

                ; System Control Block

                SCB:

                ; Expansion Area - 6 bytes
```

```

1E9C 00         hashl:          db      0
1E9D 00000000   hash:           dw      0,0
1EA1 31         version:        db      31h

                ; Utilities Section - 8 bytes

1EA2 00000000   util$flgs:      dw      0,0
1EA6 0000       dspl$flgs:      dw      0
1EA8 0000                       dw      0

                ; CLP Section - 4 bytes

1EAA 0000       clp$flgs:       dw      0
1EAC 0000       clp$errcde:     dw      0

                ; CCP Section - 8 bytes

1EAE 00         ccp$comlen:     db      0
1EAF 00         ccp$curdrv:     db      0
1EB0 00         ccp$curusr:     db      0
1EB1 0000       ccp$conbuff:    dw      0
1EB3 0000       ccp$flgs:       dw      0
1EB5 00                         db      0

                ; Device I/O Section - 32 bytes

1EB6 00         conwidth:       db      0
1EB7 00         column:                 db      0
1EB8 00         conpage:        db      0
1EB9 00         conline:        db      0
1EBA 0000       conbuffadd:     dw      0
1EBC 0000       conbufflen:     dw      0
1EBE 0000       conin$rflg:     dw      0
1EC0 0000       conout$rflg:    dw      0
1EC2 0000       auxin$rflg:     dw      0
1EC4 0000       auxout$rflg:    dw      0
1EC6 0000       lstout$rflg:    dw      0
1EC8 00         page$mode:      db      0
1EC9 00         pm$default:     db      0
1ECA 00         ctlh$act:       db      0
1ECB 00         rubout$act:     db      0
1ECC 00         type$ahead:     db      0
1ECD 0000       contran:        dw      0
1ECF 0000       conmode:        dw      0
1ED1 00                         db      0
1ED2 00                         db      0
1ED3 24         outdelim:       db      '$'
1ED4 00         listcp          db      0
1ED5 00         qflag:          db      0

                ; BDOS Section - 42 bytes

1ED6 9C1E       scbadd:                 dw      scb
1ED8 8000       dmaad:          dw      0080h
1EDA 00         olddsk:                 db      0
```

```
1EDB 0000      info:          dw      0
1EDD 00        resel:         db      0
1EDE 00        relog:         db      0
1EDF 00        fx:            db      0
1EE0 00        usrcode:       db      0
1EE1 0000      dcnt:          dw      0
1EE3 0000      searcha:       dw      0
1EE5 00        searchl:       db      0
1EE6 01        multcnt:       db      1
1EE7 00        errormode:     db      0
1EE8 00FFFFFF  searchchain:   db      0,0ffh,0ffh,0ffh
1EEC 00        temp$drive:    db      0
1EED 00        errdrv:                db      0
1EEE 0000                     dw      0
1EF0 00        media$flag:    db      0
1EF1 0000                     dw      0
1EF3 00        bdos$flags:    db      0
1EF4 FFFFFFFFFFstamp:         db      0ffh,0ffh,0ffh,0ffh,0ffh
1EF9 0000      commonbase:    dw      0
1EFB C37B00    error:                 jmp      error$sub
1EFE 0600      bdosadd:       dw      base+6

               endif
               endif

               ;      ************************
               ;      Directory Hashing Module
               ;      ************************

               ; Hash format
               ; xxsuuuuu xxxxxxxx xxxxxxxx ssssssss
               ; x = hash code of fcb name field
               ; u = low 5 bits of fcb user field
               ;     1st bit is on for XFCB's
               ; s = shiftr(mod || ext,extshf)

               if not BANKED

               hashorg:
1F00                  org     base+(((hashorg-base)+255) and 0ff00h)
               endif

               init$hash:
                      ; de = .hash table entry
                      ; hl = .dir fcb
1F00 E5D5CD4E1F       push h! push d! call get$hash
                      ; Move computed hash to hash table entry
1F05 E1119D1E01       pop h! lxi d,hash! lxi b,4

               if BANKED
                      lda hash$tbla+2! call move$out
               else
1F0C CD4BFF           call movef
               endif

                      ; Save next hash table entry address
```

```
1F0F 22D31C             shld arecord1
                        ; Restore dir fcb address
1F12 E1C9               pop h! ret

                  set$hash:
                        ; Return if searchl = 0
1F14 B7C8               ora a! rz
                        ; Is searchl < 12 ?
1F16 FE0CDA4A1F         cpi 12! jc set$hash2 ; yes - hashl = 0
                        ; Is searchl = 12 ?
1F1B 3E02CA221F         mvi a,2! jz set$hash1 ; yes - hashl = 2
1F20 3E03               mvi a,3 ; hashl = 3
                  set$hash1:
1F22 329C1E             sta hashl
1F25 EB                 xchg
                        ; Is dir hashing invoked for drive?
1F26 CDD21FC8           call test$hash! rz ; no
1F2A EB                 xchg
1F2B 3ADF1E             lda fx
1F2E FE10CA4E1F         cpi 16! jz get$hash ; bdos fx = 16
1F33 FE23CA3D1F         cpi 35! jz set$hash15
1F38 FE14D24E1F         cpi 20! jnc get$hash ; bdos fx = 20 or above
                  set$hash15:
1F3D 3E02329C1E         mvi a,2! sta hashl ; bdos fx = 15,17,18,19, or 35
                        ; if fcb wild then hashl = 0, hash = fcb(0)
                        ;                else hashl = 2, hash = get$hash
1F42 E5CD2110E1         push h! call chk$wild! pop h! jnz get$hash
                  set$hash2:
1F4A AF329C1E           xra a! sta hashl
                        ; jmp get$hash

                  get$hash:
                        ; hash(0) = fcb(0)
1F4E 7E329D1E23         mov a,m! sta hash! inx h! xchg
                        ; Don't compute hash for dir lbl & sfcb's
1F54 210000E620         lxi h,0! ani 20h! jnz get$hash6
                        ; b = 11, c = 8, ahl = 0
                        ; Compute fcb name hash (000000xx xxxxxxxxx xxxxxxxx) (ahl)
1F5C 01080B             lxi b,0b08h
                  get$hash1:
                        ; Don't shift if fcb(8)
1F5F 0DC5CA741F         dcr c! push b! jz get$hash3
                        ; Don't shift if fcb(6)
1F64 0D0DCA741F         dcr c! dcr c! jz get$hash3
                        ; ahl = ahl * 2
1F69 298FF578           dad h! adc a! push a! mov a,b
                        ; is b odd?
1F6D 1FDA751F           rar! jc get$hash4 ; yes
                        ; ahl = ahl * 2 for even fcb(i)
1F71 F1298F             pop a! dad h! adc a
                  get$hash3:
1F74 F5                 push a
                  get$hash4:
                        ; a = fcb(i) & 7fh - 20h divided by 2 if even
1F75 1AE67FD620         ldax d! ani 7fh! sui 20h! rar! jnc get$hash5
1F7E 17                 ral
```

```
                      get$hash5:
                              ; ahl = ahl + a
1F7F 4F0600                   mov c,a! mvi b,0
1F82 F109CE00C1               pop a! dad b! aci 0! pop b
                              ; advance to next fcb char
1F87 1305C25F1F               inx d! dcr b! jnz get$hash1
                      get$hash6:
                              ; ahl = 000000xx xxxxxxxx xxxxxxxx
                              ; Store low 2 bytes of hash
1F8C 229E1E219D               shld hash+1! lxi h,hash
                              ; hash(0) = hash(0) (000uuuuu) | xx000000
1F92 E6030F0FB6               ani 3! rrc! rrc! ora m! mov m,a
                              ; Does fcb(0) = e5h, 20h, or 21h?
1F98 E620C2C91F               ani 20h! jnz get$hash9 ; yes
                              ; bc = 00000mmm mmmeeeee, m = module #, e = extent
1F9D 1AE61F4F13               ldax d! ani 1fh! mov c,a! inx d! inx d
1FA3 1AE63F0F0F               ldax d! ani 3fh! rrc! rrc! rrc! mov d,a
1FAA E607477AE6               ani 7! mov b,a! mov a,d! ani 0e0h! ora c! mov c,a
                              ; shift bc right by # of bits in extmsk
1FB2 3AA91C                   lda extmsk
                      get$hash7:
1FB5 1FD2C41F                 rar! jnc get$hash8
1FB9 F5                       push a
1FBA 781F47                   mov a,b! rar! mov b,a
1FBD 791F4F                   mov a,c! rar! mov c,a
1FC0 F1C3B51F                 pop a! jmp get$hash7
                      get$hash8:
                              ; hash(0) = hash(0) (xx0uuuuu) | 00s00000
1FC4 78E6010F0F               mov a,b! ani 1! rrc! rrc
                      get$hash9:
1FC9 0FB677                   rrc! ora m! mov m,a
                              ; hash(3) = sssssss
1FCC 1103001971               lxi d,3! dad d! mov m,c! ret

                      test$hash:
1FD2 2AA21C7DB4               lhld hash$tbla! mov a,l! ora h! inr a! ret

                      search$hash:
                              ; Does hash table exist for drive?
1FD9 CDD21FC8                 call test$hash! rz ; no
                              ; Has dir hash search been disabled?
1FDD 3A9C1E3CC8               lda hashl! inr a! rz ; yes
                              ; Is searchl = 0?
1FE2 3AE51EB7C8               lda searchl! ora a! rz ; yes
                              ; hashmx = cdrmaxa if searchl ^= 1
                              ;          dir$max if searchl = 1
1FE7 2A8E1C5E23               lhld cdrmaxa! mov e,m! inx h! mov d,m
1FED EB3DC2F51F               xchg! dcr a! jnz search$h0
1FF2 2AAC1C                   lhld dir$max
                      search$h0:
1FF5 22DD1C                   shld hashmx

                      if BANKED
                              ; call search$hash in resbdos, a = bank, hl = hash tbl addr
                              lda hash$tbla+2! lhld hash$tbla! call srch$hash
                              ; Was search successful?
```

```
                    jnz search$h1 ; no
                    ; Is directory read required?
                    lda rd$dir$flag! ora a! mvi c,0
                    cnz r$dir2 ; yes if Z flag reset
                    ; Is function = 18?
                    lda fx! sui 18! rz ; Never reset dcnt for fx 18
                    ; Was media change detected by above read?
                    lda hashl! inr a! cz setenddir ; yes
                    xra a! ret ; search$hash successful
              search$h1:
                    ; Was search initiated from beginning of directory?
                    call end$of$dir! rnz ; no
                    ; Is bdos fx = 15,17,19,22,23,30?
                    call tst$log$fxs! rnz ; no
                    ; Disable hash & return successful
                    mvi a,0ffh! sta hashl
                    lhld cdrmaxa! mov e,m! inx h! mov d,m! xchg
                    dcx h! call set$dcnt$dblK1! xra a! ret
              else
1FF8 2AA21C444D     lhld hash$tbla! mov b,h! mov c,l
1FFD 2ADD1CEB       lhld hashmx! xchg
                    ; Return with Z flag set if dcnt = hashmx
2001 2AE11EE5CD     lhld dcnt! push h! call subdh! pop d! ora l! rz
                    ; Push hashmx - dcnt (# of hashtbl entries to search)
                    ; Push dcnt + 1
200B E513EBE5       push h! inx d! xchg! push h
                    ; Compute .hash$tbl(dcnt)
200F 2B292909       dcx h! dad h! dad h! dad b
              search$h1:
                    ; Advance hl to address of next hash$tbl entry
2013 1104001911     lxi d,4! dad d! lxi d,hash
                    ; Do hash u fields match?
201A 1AAE61FC2      ldax d! xra m! ani 1fh! jnz search$h3 ; no
                    ; Do hash's match?
2021 CD9E20CA67     call search$h6! jz search$h4 ; yes
              search$h2:
2027 EBE1           xchg! pop h
              search$h25:
                    ; de = .hash$tbl(dcnt), hl = dcnt
                    ; dcnt = dcnt + 1
2029 23E3           inx h! xthl
                    ; hl = # of hash$tbl entries to search
                    ; decrement & test for zero
                    ; Restore stack & hl to .hashtbl(dcnt)
202B 2B7DB4E3E5     dcx h! mov a,l! ora h! xthl! push h
                    ; Are we done?
2030 EBC21320       xchg! jnz search$h1 ; no - Keep searching
                    ; Search unsuccessful
2034 E1E1           pop h! pop h
                    ; Was search initiated from beginning of directory?
2036 CD940CC0       call end$of$dir! rnz ; no
                    ; Is fx = 15,17,19,22,23,30 & drive removeable?
203A CDB30BC0       call tst$log$fxs! rnz ; no
                    ; Disable hash & return successful
203E 3EFF329C1E     mvi a,0ffh! sta hashl
2043 2A8E1C5E23     lhld cdrmaxa! mov e,m! inx h! mov d,m! xchg
```

```
204A 2BCDD411AF        dcx h! call set$dcnt$dblk1! xra a! ret

               search$h3:
                       ; Does xdcnt+1 = 0ffh?
2050 3AE01C3CCA        lda xdcnt+1! inr a! jz search$h5 ; yes
                       ; Does xdcnt+1 = 0feh?
2057 3CC22720          inr a! jnz search$h2 ; no - continue searching
                       ; Do hash's match?
205B CD9E20C227        call search$h6! jnz search$h2 ; no
                       ; xdcnt+1 = 0feh
                       ; Open user 0 search
                       ; Does hash u field = 0?
2061 7EE61FC227        mov a,m! ani 1fh! jnz search$h2 ; no
                       ; Search successful
               search$h4:
                       ; Successful search
                       ; Set dcnt to search$hash dcnt-1
                       ; dcnt gets incremented by read$dir
                       ; Also discard search$hash loop count
2067 2AE11EEB          lhld dcnt! xchg
206B E12B22E11E        pop h! dcx h! shld dcnt! pop b
                       ; Does dcnt&3 = 3?
2071 7DE603FE03        mov a,l! ani 03h! cpi 03h! rz ; yes
                       ; Does old dcnt & new dcnt reside in same sector?
2077 7BE6FC5F          mov a,e! ani 0fch! mov e,a
207B 7DE6FC6F          mov a,l! ani 0fch! mov l,a
207F CD3D0BB5C8        call subdh! ora l! rz ; yes
                       ; Read directory record
2084 CDCC0C            call read$dir2
                       ; Has media change been detected?
2087 3A9C1E3CCC        lda hashl! inr a! cz setenddir ; dcnt = -1 if hashl = 0ffh
208E AFC9              xra a! ret
               search$h5:
                       ; xdcnt+1 = 0ffh
                       ; Make search to save dcnt of empty fcb
                       ; Is hash$tbl entry empty?
2090 7EFEF5C227        mov a,m! cpi 0f5h! jnz search$h2 ; no
               search$h55:
                       ; xdcnt = dcnt
2096 EBE122DF1C        xchg! pop h! shld xdcnt! jmp search$h25
               search$h6:
                       ; hash compare routine
                       ; Is hashl = 0?
209E 3A9C1EB7C8        lda hashl! ora a! rz ; yes - hash compare successful
                       ; b = 0f0h if hashl = 3
                       ;     0d0h if hashl = 2
20A3 4F0F0F0FF6        mov c,a! rrc! rrc! rrc! ori 1001$0000b! mov b,a
                       ; hash s field must be screened out of hash(0)
                       ; if hashl = 2
                       ; Do hash(0) fields match?
20AA 1AAEA0C0          ldax d! xra m! ana b! rnz ; no
                       ; Compare remainder of hash fields for hashl bytes
20AE E52313CDEB        push h! inx h! inx d! call compare
20B4 E1C9              pop h! ret
               endif

```

```
                        fix$hash:
20B6 CDD21FC8                   call test$hash! rz
20BA 21D91C119D                 lxi h,save$hash! lxi d,hash! lxi b,4
20C3 E5D5C5CD4B                 push h! push d! push b! call movef
20C9 2AA21CE5                   lhld hash$tbla! push h
20CD CDFE0ACD4E                 call get$dptra! call get$hash
20D3 2AE11E2929                 lhld dcnt! dad h! dad h
20D8 D119                       pop d! dad d
20DA C1D1D5C5                   pop b! pop d! push d! push b

                        if BANKED
                                lda hash$tbla+2! call move$out
                        else
20DE CD4BFF                     call movef
                        endif

20E1 C1E1D1C34B                 pop b! pop h! pop d! jmp movef

                        if not MPM
                        if BANKED

                                ds      1
                        last:
                                org     (((last-base)+255) and 0ff00h) - 1
                                db      0
                        endif

                        else
                                ds      192
                        last:
                                org     (((last-base)+255) and 0ff00h) - 192

                                ;       bnkbdos patch area

                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                                dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                                dw      0,0,0,0,0,0,0,0,0,0,0,0

                        free$root:      dw      $-$
                        open$root:      dw      0
                        lock$root:      dw      0
                        lock$max:       db      0
                        open$max:       db      0

                        ;       BIOS access table

                        bios    equ     $               ; base of the bios jump table
                        bootf   equ     bios            ; cold boot function
                        wbootf  equ     bootf+3         ; warm boot function
                        constf  equ     wbootf+3        ; console status function
                        coninf  equ     constf+3        ; console input function
                        conoutf         equ     coninf+3        ; console output function
                        listf   equ     conoutf+3       ; list output function
```

```
                punchf equu     listf+3         ; punch output function
                readerf         equ     punchf+3        ; reader input function
                homef   equ     readerf+3       ; disk home function
                seldskf         equ     homef+3         ; select disk function
                settrkf         equ     seldskf+3       ; set track function
                setsecf         equ     settrkf+3       ; set sector function
                setdmaf         equ     setsecf+3       ; set dma function
                readf   equ     setdmaf+3       ; read disk function
                writef equ      readf+3         ; write disk function
                liststf         equ     writef+3        ; list status function
                sectran         equ     liststf+3       ; sector translate

                endif

20E7                            end
```

```
ADDH            0B04 2374# 4151
ADDLIST         000D 1831  6743#
ADRIVE          1CCC  595  1725  1898  2558  2569  4085  4144  4488  6836# 6982
                     7048  7130  7180
ADVDMA          031B  670   686#
ALLOCA          1C9C 2712  2744  2761  2762  2764  2854  2866  5985  6197  6736#
ALLOCWD         1445 4149  4153#
ARECORD         1CCD 1950  1961  1961  2089  2097  2098  2098  2107  4143  4173
                     4196  4204  4208  4235  5307  6837# 7017  7018  7043  7043
ARECORD1        1CD3 2100  2874  2883  4194  6841# 7372
ARET            077A  244  1539  1656  1674# 1675  1717  4471  5262  5275  5398
                     5938  6018  6339  6656  6694
ATRAN           09B0 2091# 3852  4095  4165  5264
ATTRIBUTES      1CD5 3176  3203  3250  3374  3439  3456  5180  5501  5574  5650
                     5672  5692  5714  5794  5827  5882  5921  6842#
AUXINRFLG       1EC2  151# 7295#
AUXINSTF        FF36   95# 1492
AUXOUTRFLG      1EC4  152# 7296#
AUXOUTSTF       FF39   96# 1496
BACKSP          0616 1358# 1362
BACKUP          04C3  948# 1302  1360
BACKX           05BE 1291# 1303
BADSEEK         15C2 4368  4372  4397#
BANKED          0000   22#   65   108   230   249   291   598   606   659   764
                      781   791   796   801   805   816   820   919   936   952
                      992   999  1012  1116  1140  1153  1252  1268  1272  1293
                     1308  1320  1363  1377  1390  1395  1411  1417  1442  1446
                     1512  1526  1585  1593  1619  1640  1751  1884  1910  2306
                     2554  2578  2599  2738  2756  2841  3015  3070  3149  3175
                     3193  3208  3219  3241  3259  3272  3380  3398  3405  3429
                     3441  3692  3721  3838  3860  3932  4045  4112  4177  4199
                     4226  4277  4390  4404  4542  4553  4582  4662  4676  4687
                     4693  4872  5589  5765  5793  5846  5877  6300  6322  6338
                     6357  6439  6508  6517  6526  6620  6666  6682  6711  6801
                     6807  6856  6893  6935  6953  6973  7020  7079  7098  7113
                     7137  7150  7162  7242  7352  7365  7480  7606  7615
BASE            0000   51#   55    56    57    58  7246  7246  7252  7336  7355
                     7355  7619  7626
BDEEBDEMHL      07B0 1761# 1966  1979
BDEEBDEPHL      07B9 1766# 1943  1970
BDOSADD         1EFE  188# 2312  7336#
BDOSE           0006  223#
BDOSE1          002A  238#  241
BDOSE2          0037  226   236   239   242#  684
BDOSFLAGS       1EF3  184#  607  2739  2757  7332#
BDOSJMP         006E  257   264# 1721
BIOS            FF00   62# 6317  7645# 7646
BIOSPG          FF00   58#   62    63    75    76    77    78    79    83    84
                       85    86    87    88    89    90    91    92    93    94
                       95    96    97    98    99   100   101   102   103   104
                      105   106
BLKMSK          1CA8 1926  2104  3916  3967  4206  4236  4255  4947  6762#
BLKOFF          1CB8 2106  3885  6776# 7061
BLKOUT          06F7 1566# 1575
BLKOUT1         070B 1569  1571#
BLKOUT2         070E 1570  1573#
```

```
BLKSHF          1CA7  1927  2043  2096  2952  4943  4949  6214  6761#
BLOCKOK         141C  4133  4136#
BNKBDOSPG       FC00    55#
BOOTF           FF00    63# 7646# 7647
BUFFA           1C8C  2194  2211  2373  2622  4885  6720# 7092
BUFFNZERO       1153  3579  3582#
BYTE            0001  1690# 6707  6741  6761  6762  6763  6769  6770  6776  6777
                      6778  6783  6784  6789  6791  6793  6794  6798  6810  6814
                      6818  6822  6823  6824  6826  6828  6838  6842  6843  6849
                      6850  6854  6858  6873  6877  6878  6880  6881  6882  6883
                      6889  6895  6913  6915  6917  6932  6950  6959
CALLOPEN        184E  5502  5505  5512#
CBDOS           0310   667   681#
CBDOS1          0314   683#
CCPCOMLEN       1EAE   135# 7278#
CCPCONBUFF      1EB1   138# 7281#
CCPCURDRV       1EAF   136# 7279#
CCPCURUSR       1EB0   137# 7280#
CCPFLGS         1EB3   139# 1032  3342  6228  7282#
CDRMAXA         1C8E  1823  2407  2496  2871  6728# 7474  7500  7539
CHECKA          1C9A  2440  6735#
CHECKALLMEDIA   1736  4609  4632# 6235
CHECKFCB        0A8C  2251# 5215  5781  5787  6040  6046
CHECKMEDIA      16FC  4607  4611# 4650
CHECKMEDIA1     170F  4623# 4629
CHECKMEDIA2     172B  4625  4628#
CHECKNPR1       131F  3887  3894#
CHECKNPR10      1375  3992  3994#
CHECKNPR10A     138C  3998  4000#
CHECKNPR11      1329  3896  3903#
CHECKNPR1A      1323  3897# 3907
CHECKNPR1B      1326  3900# 3996
CHECKNPR2       133F  3908  3910#
CHECKNPR21      135E  3921  3924#
CHECKNPR23      135F  3923  3926#
CHECKNPR8       1366  3930  3975  3981#
CHECKNPR9       1367  3976  3983#
CHECKNPRS       1304  3853  3872# 4214
CHECKRODIR      0AE6  2350# 3203  3205  3412  6370
CHECKROFILE     0AE9  2354# 4062
CHECKSUM        0B46  2429# 2689
CHECKWILD       1015  3344# 3378  5489  5805  6008  6355  6501  6533
CHECKWILD0      1018  3347# 3392
CHECKWRITE      0AF7  2364# 2592  4043
CHEKFCB         0A96  2256  2281# 5706
CHEKFCB1        0AA0  2289  2295#
CHKAM1          1743  4644# 4653
CHKAM2          1758  4645  4652#
CHKEXITFXS      0BD1  1900  2518# 2695  4627
CHKINVFCB       0C84  2638# 2642  3542  4324
CHKMEDIA1       0AB5  2312  2315#
CHKMEDIA2       0AB7  2318# 2522
CHKMEDIAFCB     0AA1  2278  2302#
CHKMEDIAFLAG    0882  1875  1880#
CHKSIZ          1CB0  1915  2433  2506  4530  6767#
CHKWILD         1021  3214  3348  3351# 7398
```

```
CHKWILD1        1023  3353# 3355
CLOSE           1104  3529# 3730  4269  4345  5698  5706
CLOSE1          1112  2262  3541# 4016  5702
CLOSEFCB        1133  3557  3562#
CLPERRCDE       1EAC   131#  650  6652  6653  7274#
CLPFLGS         1EAA   130# 7273#
CLREXT          0B18  2392# 4577  5744  5797
CLRMODNUM       0B12  2387# 5470  5744  5798
COLUMN          1EB7   144#  891   943   970  1015  1044  1047  1051  1078  1083
                      1100  1127  1191  1220  1239  1292  1355  7288#
COMMONBASE      1EF9   186# 6330  7101  7114  7334#
COMPARE         07EB  1793# 1796  2558  2569  4086  4773  4864  5015  5133  5303
                      7032  7174  7592
COMPAREEXTENTS  11BF  3606  3648#
COMPAREMODEXT   1BB9  6377  6418#
COMPARERR       1613  4444# 4469  6366
COMPCDR         0B27  2404# 2416  3005  4629
COMPCOL         0715   871  1193  1241  1352  1356  1361  1582#
COMPEXT         0E1C  2914# 3051  3727  4330  6425
COMPOUT         0472   871   883   888#
COMPUTECS       0A76  2192# 2208# 2439
COMPUTECS0      0A7C  2196# 2203  2213# 2215  2226  2235
COMPUTECS1      0A7F  2199# 2201
COMPUTERR       15E0  4426# 4465  4481  6364
CONB0           03FB   809   829# 1521
CONB1           0416   761   843# 1517
CONB2           0419   839   840   846#
CONB3           041C   828   848# 1283
CONBRK          03B0   776# 1535
CONBRK1         03D2   778   790   808#
CONBRK2         03E0   745   811   814#  828
CONBRK3         03F1   823   826#
CONBRKX         03BD   786#  878
CONBUFFADD      1EBA   147#  996  1009  1111  1452  7291#
CONBUFFLEN      1EBC   148# 1009  7292#
CONECH          035E   734# 1456
CONECHO         0362   736#  745   747   748
CONECH1         0372   738   742#
CONECH2         0381   744   746#
CONIN           0353   727#  738   996  1010  1488
CONINF          FF09    69#   77#  731   804   819  1521  7649# 7650
CONINRFLG       1EBE   149# 7293#
CONLINE         1EB9   146# 7290#
CONMODE         1ECF   160#  774   824   834   855   875   929  1401  1510  1550
                      1551  7304#
CONOUT          0449   868#  916   929   932   942   966   972   978   978  1048
                      1052  1222  1453
CONOUTF         FF0C    70#   78#  850   863   866   956   960  1062  1374  1479
                      1577  7650# 7651
CONOUTRFLG      1EC0   150# 7294#
CONOUTSTF       FF33    94#
CONPAGE         1EB8   145# 7289#
CONSTF          FF06    68#   76#  762   795  1519  7648# 7649
CONSTX          039B   760#  777   782  1482  1484
CONTRAN         1ECD   159# 7303#
CONWIDTH        1EB6   143# 1015  7287#
```

```
COPYALV         0D13  2734# 2847  2880  6246
COPYDIR         0FDF  3318# 3421
COPYDIR0        0FE1  3322# 3710
COPYDIR1        0FE5  3325# 3329
COPYDIR2        0FF8  3323  3330# 3455
COPYDIRLOC      0E16  2907# 3237  3430  3433  3465  6011  6375
COPYSTAMP       1C42  6592  6607  6610#
COPYUSERNO      1030  3357# 3388  3403  3431
CR              000D    46#  477   546   754   906   978  1150  1453  5402
CRLF            04E8   594   626   967   976# 1103  1271
CRLFP           04D0   963# 1313  1330  1336
CRLFP0          04D8   969#  973
CSEARCH         18CE  5738# 5771
CSEARCH1        18E0  5740  5742#
CSEARCH2        18EF  5743  5745#
CSEARCH3        18F4  5741  5748#
CSEARCH4        190B  5749  5755#
CTL             005E    48#  916
CTLA            0001    28#  953  1068  1188
CTLB            0002    29# 1205
CTLC            0003    30#  823   836  1403  1517  1521
CTLCSTATRET     06C5  1515  1524#
CTLE            0005    31# 1264  1425
CTLF            0006    32# 1069  1154  1422
CTLG            0007    33# 1229  1378  1396
CTLH            0008    34#  757   900   960  1052  1222  1235
CTLHACT         1ECA   156# 1236  7300#
CTLK            000B    35# 1224
CTLOUT          048F   911# 1080  1345  1393
CTLP            0010    36#  748   840  1279
CTLQ            0011    37#  747   827   839
CTLR            0012    38# 1318
CTLS            0013    39#  744   790   809
CTLU            0015    40# 1307
CTLW            0017    41# 1070  1161  1185  1208  1330  1421
CTLX            0018    42# 1288
CTLZ            001A    43#
CURBCBA         1CFC  2601  6951# 7024  7065  7130  7135  7174  7176  7205  7208
                      7210  7214  7222
CURDMA          1C8A   203# 2003  2615  6714# 7024  7089
CURDSK          1CCB  1713  2332  2345  4504  4648  5011  5066  5220  5335  5347
                      5353  5359  5361  5380  5436  6086  6106  6111  6113  6125
                      6130  6135  6832# 6863# 7048
CURRECA         1C92  1825  1852  1959  1990  6730#
CURSELECT       1680  2532  4503# 4598  4654  5741  5984  6015  7054
CURTRKA         1C90  1824  1851  1957  1987  6729#
DCNT            1EE1   175# 1933  2406  2533  2629  2648  2659  2668  2668  2677
                      3044  3140  3164  3660  4621  4630  4883  5079  5751  5752
                      5754  6908# 7321# 7506  7559  7560  7602
DEBLOCK         1D05  2604  6974  6999  7003#
DEBLOCK1        1D35  7030  7034#
DEBLOCK10       1DDA  7036  7123  7132# 7159
DEBLOCK11       1DDD  7134#
DEBLOCK15       1D42  7037  7039#
DEBLOCK2        1D6A  7028  7038  7055#
DEBLOCK25       1D7E  7059  7062#
```

```
DEBLOCK3         1D88  7061  7068#
DEBLOCK35        1D8B  7067  7073#
DEBLOCK4         1D8E  7075#
DEBLOCK45        1D96  7032  7084#
DEBLOCK5         1DA0  7087# 7088
DEBLOCK6         1DB4  7091  7093#
DEBLOCK8         1DCA  7016  7126#
DEBLOCK9         1DD1  7026  7045  7076  7129# 7174
DEBLOCKDTA       1D02  3869  4220  6264  6970#
DEBLOCKIO        1DE2  7050  7074  7144#
DELETE           0F63  3171# 5776
DELETE00         0F6C  3190# 3229
DELETE02         0F7E  3210  3214  3223  3224  3228#
DELETE1          0F84  3191  3233#
DELETE10         0F87  3236# 3279  4738  5892
DELETE11         0F8A  3217  3238# 3400
DELETE12         0F8D  3243  3253#
DELIM            0273   422   437   471   515   548#
DELIM1           0278   551#  556
DELIM2           0283   552   558#
DELIM3           028F   566   568#
DELIMITERS       0265   546#  550
DEVINITF         FF3F    98#
DEVTBLF          FF3C    97#
DIOCOMP          0871  1865  1872# 6237  6261
DIRBCBA          1C9E  2547  2597  4175  6737#
DIRBLK           1CAE  2865  6766#
DIRCNT           1CBD   227  3887  3889  3996  6783#
DIRINP           0683  1476  1483#
DIRINP1          0688  1478  1487#
DIRLOC           1CC2  2911  2969  3077  6791#
DIRMAX           1CAC  2495  2667  6765# 7476
DIRREC           0004  2014# 2016
DIRSTAT          067A  1477  1480#
DIRTOUSER        0C5F  2617# 5757
DISCARD          0C19  2544  2549#
DISCARD0         0C1B  2541  2551#
DISCARDDATA      0C10  2462  2543# 2837  6252
DISCARDDATABCB   0C08  2540# 4196  4231
DISCARDDIR       0C16  2546# 2838  4605  4620
DISKF            000C   314#
DISKREAD         12B4  3808# 4416
DISKREAD0        12BF  3818# 3824  3846
DISKSELECT       1666  4486# 4508  4648  5459  6030
DISKSELECT1      1669  4489# 7048
DISKWR1          144F  4105  4161#
DISKWR10         1453  4099  4166#
DISKWR11         14B1  4167  4168  4211#
DISKWR2          14EF  4245  4261#
DISKWRITE        13AD  4039# 4423
DISKWRITE0       13DD  4070  4073#
DISKWRITE15      13F6  4086  4089#
DISKWRITE2       13FB  4083  4106#
DISKWRU          144D  4152  4158#
DLOG             1C88  1892  2329  4492  4643  5425  5432  5961  6069  6070  6239
                       6700#
```

```
DMAAD          1ED8   168#  687   690   695  1110  2608  5451  5979  7094  7314#
DMINX          1CC4  2087  2942  3508  3938  4122  4148  6399  6794#
DMPOS0         0969  2045# 2046
DMPOS1         0977  2052# 2056
DMPOS2         0980  2055  2057#
DMPOSITION     0962  2041# 2086  3935  4121  6390
DMSET          116F  3584  3595#
DOCTLH         0565  1237# 1250  1254  1396
DOCTLU         05D4  1312# 1322
DONTWRITE      14E2  4216  4221  4241#
DORUBOUT       0588  1236  1251#
DPBADDR        1C98  1830  1833  6016  6734# 6743
DPBLIST        0011  1835  6771#
DPTR           1CD8  2373  2533  2682  6889#
DREC           1CB6  1935  2433  2442  6775#
DRVLBL         0E0A  2892  2903#
DRVLBLA        1C94  1826  2861  2905  4685  4898  6474  6731#
DRVRELOG       0BF1  2531# 4627
DRVTBLF        FF42    99#
DSKERR         0727   595  1630#
DSKMAP         0010  2029# 2070  2773  2775  3358  3421  3567  3568  4146
DSKMSG         0719   596  1629#
DSKMSK         0003  2016# 2629  2677
DSKSHF         0002  1934  2015#
DSPLFLGS       1EA6   126# 7268#
DTABCBA        1CA0  2541  2544  4660  6738# 6971  7000
ECHOC          038C   738   751#  913
EFCB           1C85  3682  6698#
EMPTY          00E5  2010# 2889  3001  3013  3257  6383  6698
ENDDIR         FFFF  1689# 2659
ENDOFDIR       0C94  2645# 2820  2880  2997  5541  5820  5871  7495  7534
ENDSEARCH      0F13  3024  3067#
ENTSP          02CB   247   654# 1659# 6347  6693
ERRDRV         1EED   182# 1725  7328#
ERRFLG         02A6   285   592#
ERROR          1EFB   187# 1726  7335#
ERRORMODE      1EE7   179#  286  1726  6188  7325#
ERRORSUB       007B   281# 7335
ERRTBL         0170   283   336#
EXTMSK         1CA9  2158  2918  2955  4905  6763# 7447
EXTNUM         000C  2019# 2133  2904  2975  3029  3161  3395  3421  3454  3493
                     3523  3601  3615  4328  4356  4431  4788  4816
EXTVAL         1CC8  2051  2160  6824#
FALSE          0000  1688# 1842  2996  4040  4421  4624
FCBDSK         1CE3  4590  6665  6917#
FCBLEN         0020  2013# 2014  2030  2031  2775  3568  3601  3703  3710
FCBNZERO       114D  3575  3578#
FCBSHF         0005  2017# 2678
FILEEXISTS     197D  3396  5822  5935#
FILL0          147C  4190# 4191  4192
FILL1          148C  4195# 4206
FIXHASH        20B6  3278  3424  3714  6383  6483  7596#
FLUSH          1AB4  5711  6259#
FLUSH0         1A82  6238# 6351
FLUSH1         1A87  6240# 6256
FLUSH3         1AA0  6243  6247#
```

```
FLUSH35         1AAA  6246  6251#
FLUSH4          1AAD  6250  6253#
FLUSH5          1AAF  6241  6255#
FLUSHF          FF48   101# 6236  6260
FLUSHX          1ABA  3999  6248  6262#
FREEMODE        1CC3  5141  6120  6123  6792#
FUNC1           035E   311  1456#
FUNC10          051D   313  1505#
FUNC100         0F26   329  6441# 6487#
FUNC101         1BEE   329  6492#
FUNC102         1BF7   329  6498#
FUNC103         0F26   329  6488# 6528#
FUNC104         1C25   330  6585#
FUNC105         1C32   330  6595#
FUNC106         06CE   330  6622# 6634#
FUNC107         1C47   330  6638#
FUNC108         1C50   331  6648#
FUNC109         06D4   331  1547#
FUNC11          06A0   313  1508#
FUNC110         06E2   331  1553#
FUNC111         06EE   331  1560#
FUNC112         06EE   332  1561#
FUNC12          1812   315  5413#
FUNC13          1818   315  5422#
FUNC14          182B   315  5455#
FUNC15          1835   315  5468#
FUNC16          18B8   316  5687#
FUNC17          18CC   316  5735#
FUNC18          191D   316  5762#
FUNC19          1928   316  5773#
FUNC2           04A0   311  1459#
FUNC20          192E   317  5778#
FUNC21          1937   317  5784#
FUNC22          1940   317  5790#
FUNC23          1989   317  5954#
FUNC24          198F   318  5959#
FUNC25          1995   318  5963#
FUNC26          199B   318  5967#
FUNC27          19A2   318  5982#
FUNC28          0B91   319  5989# 5998#
FUNC29          19AB   319  6002#
FUNC3           0664   311  1462#
FUNC30          19B1   319  6006#
FUNC31          19BD   319  6013#
FUNC32          19C7   320  6020#
FUNC33          19DB   320  6037#
FUNC34          19E4   320  6043# 6145
FUNC35          19ED   320  6049#
FUNC36          1655   321  6054#
FUNC37          19F3   321  6056#
FUNC38          06CE   321  6090# 6140#
FUNC39          06CE   321  6117# 6141#
FUNC40          19E4   322  6145#
FUNC42          06CE   322  6150# 6157#
FUNC43          06CE   322  6152# 6158#
FUNC44          1A17   323  6162#
```

```
FUNC45          1A25   323  6175#
FUNC46          1A2A   323  6193#
FUNC47          1A67   323  6223# 6227#
FUNC48          1A79   324  6234#
FUNC49          1AC4   324  6268# 6272#
FUNC49SET       1ADF  6276  6279#
FUNC50          1AEA   324  6286# 6290#
FUNC50RET       1B0F  6297  6336#
FUNC6           066A   312  1474#
FUNC7           068E   312  1491#
FUNC8           0694   313  1495#
FUNC9           069A   313  1499#
FUNC98          1A82   328  6351#
FUNC99          1B16   328  6353#
FUNCRET         06CE  1540# 1657# 4869  6140  6141  6149  6157  6158  6223  6268
                      6286  6634
FUNCTAB         00EC   257   310#  314   325
FUNCTAB2        0152   263   327#  334
FWFMSK          0080  2026# 2400  3540  4030
FX              1EDF   173#  226   609   669   682   702   863   878   926   937
                      1126  1569  1737  1896  2169  2509  4167  4322  4626  4727
                      4801  6243  7319# 7390  7489
FXSTSMSG        0760   344  1644#
GETALLOCBIT     0CEA  2698# 2720  3294  3301  3958
GETATTS         09D6  2109# 3173  3374  3438  5493  5692  5794
GETATTSLOOP     09DF  2115# 2119
GETBLOCK        0FA0  3281# 4131
GETBUFFA        0C2B  2574# 2611  7024
GETDE0          0E35  2932# 2955
GETDE1          0E38  2936# 2939
GETDE2          0E42  2938  2941#
GETDE3          0E4F  2944  2946#
GETDIREXT       0E2D  2151  2926# 3482  3548  3602  3796  4332  5822  6393
GETDIRMODE      176D  3210  3223  4684# 4696  5592  5850  5880  6496  6532
GETDM           098A  2072# 2088  3941  3944  4127
GETDMA          0982  2069# 2074  2639  6433  6504
GETDMD          0998  2077  2079#
GETDPTRA        0AFE  2352  2369# 2771  2882  3009  3195  3239  3337  3477  3637
                      4464  4706  4815  4830  4869  6363  6383  6415  7601
GETDTBA         17A7  3706  4850  4878# 5910  6505
GETEXTA         09FD  2124  2131# 2157  2177  2394  2957  3559  3725  4906  4916
                      5743  6454  6521  6535  6559  6676
GETFCB          0A16  2146# 3785  3817  4068  5261  6390
GETFCB0         0A2B  2150  2152#
GETFCB1         0A32  2153  2155#
GETFCBA         0A0D  2140# 2148  2165  2929  3460  5542  6395
GETFCBADDS      1BAF  3567  6400  6414# 6419
GETFILESIZE     1628  4455# 6052
GETHASH         1F4E  7361  7391  7393  7398  7403# 7601
GETHASH1        1F5F  7411# 7433
GETHASH3        1F74  7413  7415  7422#
GETHASH4        1F75  7419  7424#
GETHASH5        1F7F  7426  7428#
GETHASH6        1F8C  7407  7434#
GETHASH7        1FB5  7448# 7453
GETHASH8        1FC4  7449  7454#
```

```
GETHASH9        1FC9   7441  7457#
GETMODNUM       0B09   2380# 2389  2398  3539  3790  4028  4035  4914  5697
GETNALBS        0D86   2743  2760  2807# 2852  6198
GETRCNTA        0A05   2136# 2142  4022  4268
GETRRA          09F5   1778  1788  2126# 4458  5674  6366
GETS1           09EF   2122# 2240  2247  2277  2538
GETSIZE         1634   4461# 4472
GFC             023F    446   463   501   515#
GFC1            0258    525   528#
GOBACK          1C64    253  1740  2519  5941  6657#
GOBACK0         1C5E   1738  1739  6655#
GOERR           078D   1702  1706  1715# 1877  1878  1904
GOERR1          0793   1724# 5943
GSP1            1A37   6202# 6207
GSP2            1A38   6203# 6205
GSP3            1A3C   6204# 6204
GSP4            1A44   6203  6206#
HASH            1E9D    120# 7262# 7363  7405  7437  7437  7514  7598
HASHL           1E9C    119# 2467  7261# 7385  7395  7400  7469  7491  7499  7538
                       7570  7583
HASHMX          1CDD    196# 6899# 7478  7504
HASHORG         1F00   7354# 7355
HASHTBLA        1CA2   2874  6739# 7366  7463  7482  7482  7503  7600  7607
HIGHEXT         1CD7   2282  4011  4050  4540  4568  4571  5479  5497  5532  5551
                       5568  5580  5922  6676  6687  6854#
HIGHFXS         005C    255   259#
HLROTL          0ABC   2322# 2336  5336  5994
HLROTL0         0ABD   2325# 2326
HLROTR          0A69   1934  2006  2184# 2347  2709  2810  2953  4944  5351
HLROTR0         0A6A   2187# 2190
HOLDDMA         030E    679#  690   693
HOME            084C   1847# 2815  2869  2981  4622  5480
HOMEF           FF18     85# 1849  7654# 7655
INCRPDCNT       1CC2   5148  6120  6132  6790#
INCRRR          07CA    669  1777# 5239  5250
INDEX           099E   2084# 3842  4082  5263
INDIC0          107B   3452# 3466
INDIC1          1095   3456  3462#
INDICATORS      1071   3436# 6010
INFO            1EDB    170#  224   267   381   622   684  1110  1128  2070  2112
                       2128  2133  2138  2226  2294  2309  2384  2965  3214  3333
                       3346  3358  3478  3681  3682  3689  4061  4146  4297  4551
                       4589  4601  4709  4712  4726  4737  4741  4810  4813  4830
                       5344  5373  5373  5425  5535  5555  5577  5768  5971  6062
                       6106  6121  6332  6340  6416  6452  6457  6665  7316#
INITHASH        1F00   2885  7358#
INITIAL0        0DA3   2856# 2859
INITIAL2        0DCE   2878# 2887  2890  2901
INITIAL3        0E04   2893  2899# 2905
INITIALCS       0B84   2446  2481#
INITIALIZE      0D90   2824# 4521
INVIS           000A   2037#
KBCHAR          0717    206#  729   761   778   790   811   830  1517  1522  1587#
LAST            1E6D   7245# 7246  7618# 7619  7625# 7626
LASTBLOCK       1CBB   2596  3868  4143  6779#
LASTDRIVE       1CBA   4085  4144  6778#
```

CP/M - PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
LASTOFF         1CB9  4088  4143  4186  6777# 7061
LEFTTST         0FBA  3297# 3315
LF              000A    47#  755   978  1151  5402
LINELEN         05E4  1194  1203  1246  1333#
LINFO           1CC3   243  4593  6022  6250  6793#
LISTCP          1ED4   162#  856   886  7308#
LISTF           FF0F    71#   79#  312   886  1570  7651# 7652
LISTSTF         FF2D    92# 7661# 7662
LOCKUNLOCK      1CC1  5224  5235  5244  6151  6153  6787#
LOGFXS          1CE4  2507  6926#
LRET            077A  1675# 2263  2628  3078  3531  3732  3834  4019  4072  4348
                      4362  4812  4922  5709
LRETEOFF        0F26   261   262   322  1482  2525  3091# 3646  5580  5892  5938
                      6163  6164  6361  6367  6368  6454  6487  6488  6532
LSNADD          1C96  1827  2277  2485  2516  2537  2846  4524  6732#
LSTACK          00EC   250   304#
LSTFCB          001F  2030#
LSTOUTRFLG      1EC6   153# 7297#
LSTREC          007F  2011# 4069
MAKE            11DD  3675# 3761  4370  5864  5889  6461  6552
MAKE0           120E  3704# 3705  3707
MAKE3A          196C  5880  5882  5884  5910  5916#
MAKEA0          1960  5820  5823#
MAKEFCBINV      0C79  2631# 3645
MAKEFLAG        1CC1  5474  5678  5830  5839  5841  5921  6788#
MAXALL          1CAA  1837  2798  2808  3289  3954  6210  6764#
MAXEXT          001F  2020# 2922  2958  3733
MAXMOD          0040  2024#
MD              0024  1621# 1625# 1630  1631  1632  1633  1634  1641  1645  1648
MEDIACHANGE     0B70  1893  2461#
MEDIAFLAG       1EF0   183# 4639  7330#
MERGE0          113F  3569# 3597
MERGED          115A  3570  3585#
MERGERR         11B9  3542  3583  3590  3592  3643#
MERGEZERO       10E6  3511# 3587  3588
MODNUM          000E  2022# 2384  3031  4325  4436  6419
MOVE            07F5   623   624  1027  1039  1786  1802# 1831  1835  3455  3480
                      3672  4708  4833  4865  5068  5079  5327  5675  6409  6509
                      6511  6611  6646  7046  7077  7160
MOVE0           07F6  1806# 1810
MOVEF           FF4B   102#  211   212  2625  2745  2746  6333  7368  7599  7609
                      7612
MOVEOUT         FF4B   193#  211# 7366  7607
MOVETPA         FF4B   194#  212# 7101  7103  7107  7121
MPM             0000    21# 1636  1651  1668  1719  1731  1742  1855  1923  1937
                      2206  2253  2284  2297  2450  2464  2475  2619  2812  2829
                      2989  3040  3103  3109  3142  3197  3244  3373  3414  3448
                      3533  3544  3632  3662  3668  3751  3765  3775  3781  3812
                      3823  3844  4008  4052  4075  4091  4108  4138  4249  4266
                      4352  4374  4382  4510  4529  4560  4764  4775  4854  4926
                      5416  5424  5442  5457  5472  5491  5508  5520  5526  5550
                      5573  5583  5627  5691  5713  5801  5808  5814  5826  5866
                      5873  5919  5940  5946  5969  5987  6028  6059  6072  6081
                      6088  6147  6167  6177  6221  6266  6284  6587  6599  6613
                      6624  6640  6661  6672  6702  6718  6796  6816  6830  6847
                      6861  6904  6919  6930  6945  7241  7614
```

```
MRGRC1          11A0  3620  3624  3628#
MRGRC2          11A1  3622  3625  3630#
MULTCNT         1EE6   178#  236   665   699  4096  5205  5227  5232  5241  5272
                      6099  6165  6849# 7324#
MULTFXS         0094   237   289#
MULTIO          02E4   666#  671
MULTIOF         FF45   100# 4005
MULTNUM         1CBE   667  3907  3991  6784#
NAMLEN          000F  2027# 2973  3168  3703  5536  5747
NEWCHECKSUM     0B44  2427# 2593
NFUNCS          0033   255   325#
NFUNCS2         000F   262   334#
NOPBLOCK        140F  4124  4129#
NORMALSTATUS    06C8  1510  1533#
NOSELECT        16E6  4594  4597#
NOSELECT0       16F0  4602# 5741
NOTBACKSP       0489   900   904#
NOTC            0654  1401  1405  1409#
NOTE            05A8  1264  1277#
NOTH            057C  1235  1247#
NOTP            05B8  1279  1286#
NOTR            0623  1159  1186  1318  1367#
NOTRUB          0593  1249  1262#
NOTU            05DB  1307  1316#
NOTX            05CF  1288  1304#
NOUPDATE        14F7  4263  4267  4273#
NOWRITE         0AD8  2340# 2366  2459  3537  4724  4734  4898  5609
NXTREC          0020  2031# 2032  2143  3479  4316  4480
OFF             0000    20#   21    22
OFFSET          1CB2  1983  6768#
OLDDSK          1EDA   169#  228  5440  5465  7315#
OLOG            1E90   112# 2502  5685  7248#
ON              FFFF    19#
OPEN            10A1  3468# 4361  5513  5812
OPEN1           10A4  3471# 5537
OPENCOPY        10A5  3474# 3773
OPENMOD         1258  3738#
OPENREEL        1233  3717# 3832  4071
OPENREEL0       125F  3736  3748#
OPENREEL1       1278  3756  3771#
OPENREEL2       127B  3770  3779# 3805
OPENREEL3       1299  3728  3794#
OPENREEL4       12AB  3798  3803#
OPENRERR        1288  3745  3759  3763  3788#
OPENUSERZERO    1868  5481  5533#
OPENX           187C  5513  5538  5540#
OPENX0          18A7  5553  5555  5557  5559  5568  5578  5581#
OPENX2          18AC  5683# 5920  5930
OPENXA          188C  5542  5544#
OUTDELIM        1ED3   161#  983  1554  7307#
PAD             0292   398   400   402   404   573#  576
PAGEMODE        1EC8   154# 7298#
PARSE0          0196   392#
PARSE6          01DC   441#  447
PARSE8          01FA   460#  464
PARSEDRV        01BE   421#
```

```
PARSENAME       01D2  417   435#
PARSEOK         0206  423   438   448   466#  511
PARSEPW         0221  445   462   493#
PARSEPW1        022B  500#  502
PARSETYPE       01F0  443   452#
PASSMSG         0760  343  1638#
PATCHFLGS       1E94 7251#
PCTLH           04CB  950   958#
PERMSG          072B  337  1631#
PERROR          021B  484#  519   522
PERROR1         021C  425   427   486#
PHYMSK          1CB5 3886  4174  6263  6770# 7127  7181  7237
PHYOFF          1CFB 6950# 7017  7086  7216  7236
PHYSHF          1CB4 2006  6769#
PMDEFAULT       1EC9  155# 7299#
PRINT           04F2  601   642   981#  988  1503
PUNCHF          FF12   83#  312  7652# 7653
QCONIN          0502  990# 1141  1145
QCONIN1         0513 1006  1008#
QCONOUTF        0439  861#  880
QDIRFCB1        17D5 4895  4901# 5594  5848  5884
QFLAG           1ED5  163#  768  1513  7309#
QSTAMP          17C7 4893# 5684  5931
QSTAMP1         17CB 4896# 4912
RANCLOSE        1577 4324  4326  4330  4343#
RANDISKREAD     15CE 4412# 6041
RANDISKWRITE    15D7 4419# 6047
RANREC          0021 2032# 2128  4298  4482
RCOUNT          1CC7 2156  2173  3819  3923  4022  4244  4268  6823#
RDBUFF          085F 1861# 3864  7157
RDDIR           0C36 2584# 2686  2696  5753
RDECHO          0634 1371  1375#
RDECH1          0638 1261  1386#
RDIR            0CAA 2662  2664# 4624
RDIR1           0CDC 2662  2693#
RDIR2           0CD9 2691# 7487
READ            052D 1109  1114# 1296  1299  1315
READDEBLOCK     12F6 3855  3867#
READDIR         0CA4 2661# 2818  2879  2996
READDIR0        0CBB 2674#
READDIR1        0CC2 2679# 2680
READDIR2        0CCC 2684# 2692  7568
READEN          0657 1150  1151  1440#
READERF         FF15   84# 1464  7653# 7654
READF           FF27   90# 1864  7659# 7660
READFSW         1CD6 1863  1870  1902  6843#
READI           051D 1102# 1108# 1505
READNO          054A 1138# 1221  1276  1331  1353  1366  1374  1419  1435
READNX          0548 1064  1135# 1156  1175  1192  1227  1230  1240  1257  1285
                     1414  1438
REBOOTE         02BC  287   645#
REBOOTX         02C2  647#  824  1407
REBOOTX0        02C5  646   649#
REBOOTX1        02C8  311   651# 6229  6231
RECCNT          000F 2028# 2138  2143  3493  3523  3615  4464  6363
RECORDOK        12D8 3821  3835#
```

```
RECORDOK1       12DE 3845  3850#
RECSIZ          0080 2012# 2014  2210  2624
RELOG           1EDE  172# 1908  2467  2529  2688  2694  4625  7318#
RENAME          103A 3361# 5957
RENAME0         105E 3419# 3426  3431
RENAME1         106E 3427#
REP0            05EC 1339# 1347
REP1            0607 1340  1348#
RESBDOSPG       FD00   56#  192   193   194   195   196   197   198   199   200
                      202   203   204   205   206   207
RESEL           1EDD  171#  245   620  4588  6659  7317#
RESELECT        16AC 4548# 5215  5473  5689  5780  5786  6039  6045  6051
RESELECT1       16C3 4546  4578#
RESELECTX       16A5 4539# 5486  5746  5775  5799  5956  6009  6354  6451  6500
                     6530
RESET37X        19F4 2473  5438  6067#
RESETDMA        032C  692#  703
RESETDMA1       032F  687   694#
RESETFWF        139A 4027# 4264  4281  4287
RESETRELOG      08AE 1898  1906#
RESETRR         07E5  702  1790# 5241  5254
RESTORERC       10F4 3519# 3554  3804  4340
RESTORERC1      1102 3524  3526#
RETBLOCK        0FCE 3295  3302  3306#
RETBLOCK0       0FD6 3291  3312#
RETMON          1C7B 6659  6692#
RETSELECT       084A 1840  1843#
RIGHTTST        0FA2 3288# 3298  3305
RLOG            1E92  113# 2314  2503  4513  4531  5360  5434  6082  6082  6704#
                     7249#
RMF             1CC1 3758  3801  3810  3921  3952  3975  3998  4040  6789#
RODERROR        077C 1700# 2367
RODMSG          0751  338  1634#
RODSK           1C86 2342  2493  5361  5432  6004  6070  6079  6699#
ROFERROR        0781 1704# 2358
ROFILE          0009 2036# 2361
ROFMSG          0742  339  1633#
ROTEST          0AF0 2356  2360# 5557
ROTL            0CFE 2715# 2715
ROTR            0D0C 2726# 2730  3309
RSEEK           1500 4292# 4415  4422  5259  6361
RSEEK2          156C 4333  4334  4336#
RSEEK3          15B7 4322  4349#
RTNPHYERRS      07A0 1720  1729#
RUBOUT          007F   44#  892  1249  1253  1397
RUBOUTACT       1ECB  157# 1250  7301#
RWFXS           1CF1 2522  6927#
RXFCB2          1C1E 6505  6514#
SAVEDCNTPOS     0F5B 3133  3139#
SAVEDCNTPOS0    0F50 3072  3132#
SAVEDCNTPOS1    0F53 3013  3087  3134#
SAVEDCNTPOS2    0F44 3054  3123#
SAVEDCNTPOSRET  0F61 3138  3146#
SAVEDMA         0325  664   689#
SAVEHASH        1CD9 6891# 7598
SAVEMOD         1CC9 3735  3790  6825#
```

```
SAVEPOS         0718  1129  1207  1213  1266  1319  1345  1591#
SAVERANR        1CD0  1788  6840#
SAVERR          07D5   664  1783# 5232
SAVERR1         07D9  1785# 1791
SAVERR2         07DE  1784  1787# 1791
SCANDM0         0D58  2776# 2805
SCANDM1         0D6B  2782  2788#
SCANDM2         0D71  2787  2794#
SCANDM3         0D80  2797  2802#
SCANDMA         0D4E  2750  2763  2766# 2898
SCANDMAB        0D2D  2748# 3273  3275  6379  6411
SCANDMB         0D32  2753# 3640
SCB             1E9C   115# 6275  7257# 7313
SCBADD          1ED6   167# 7313#
SCBPG           FB00    57#   67    68    69    70    71   112   113   115   119
                       120   121   125   126   130   131   135   136   137   138
                       139   143   144   145   146   147   148   149   150   151
                       152   153   154   155   156   157   158   159   160   161
                       162   163   167   168   169   170   171   172   173   174
                       175   177   178   179   180   181   182   183   184   185
                       186   187   188
SCFXS           1CF8  2525  6928#
SCONOUTF        FF0C   865  1577#
SEARCH          0E8D  2973  2976# 5759  6453  6458
SEARCH1         0E90  2979# 3395
SEARCHA         1EE3  2965  2999  3160  3389  3395  5766  5768  6803# 7322#
SEARCHCHAIN     1EE8   180# 7326#
SEARCHEXT       0EF4  3030  3035#
SEARCHEXTNUM    0E8B  2974# 3188  3234  3410  3445  4460  4812  6373  6550
SEARCHFIN       0F20  2995  2997  3007  3084#
SEARCHH0        1FF5  7475  7477#
SEARCHH1        2013  7484  7493# 7512# 7530
SEARCHH2        2027  7519# 7546  7548  7552  7576
SEARCHH25       2029  7521# 7579
SEARCHH3        2050  7516  7542#
SEARCHH4        2067  7518  7554#
SEARCHH5        2090  7544  7572#
SEARCHH55       2096  7577#
SEARCHH6        209E  7518  7548  7580#
SEARCHHASH      1FD9  2995  7465#
SEARCHI         0E73  2963# 2978  3168  3685  5536  5760
SEARCHI1        0E79  2966# 3161  3395
SEARCHL         1EE5   177# 2967  3010  5760  6798# 7323# 7471
SEARCHLOOP      0ECF  3017  3018  3023# 3066
SEARCHMOD       0F1D  3031  3082#
SEARCHN         0E99  2984# 3019  3062  3073  3098  3099  3100  3127  3161  3168
                      3229  3279  3425  3464  3685  4470  5536  5760  6387
SEARCHNAMLEN    0E86  2972# 3470  3564  3755  4714  4919  5891  6503  6539  6571
SEARCHNEXT      0EC0  3002  3008#
SEARCHNM        0F2D  3033  3097#
SEARCHOK        0F0C  3020  3025  3027  3034  3063# 3107
SEARCHUSER0     1CE1  3057  3100  4587  5506  5514  6913#
SECTOR          1D00  2001  6964# 6992  6999
SECTPT          1CA5  1834  1966  1970  1979  6760# 6771
SECTRAN         FF30    93# 1999  7662#
SEEK            08CA  1954# 3858  4197  4224  7148
```

```
SEEK0           08E0  1962# 1968
SEEK1           08F7  1965  1969# 1976
SEEK2           090E  1973  1977#
SEEKCOPY        0C3E  2589# 3326  3463  3641  4724  4851  5609  5912  6484  6573
SEEKDIR         08B3  1920# 2586
SEEKERR         15C9  4313  4348  4402#
SEEKOK          15B8  4362  4379#
SEEKOK1         15BD  4342  4388#
SEEKOK2         15BD  4329  4394#
SELDSK          1CC6   228  1898  2493  4501  4504  4596  5461  5465  5965  5993
                      6818# 6822#
SELDSKF         FF1B    86# 1818  7655# 7656
SELECT          168D  4504  4507#
SELECTDISK      07FF  1812# 4494
SELERROR        0786  1708# 4495
SELMEMF         FF51   104#
SELMSG          0734   340  1632#
SEQDISKREAD     12B4  3807# 5782
SEQDISKWRITE    13AD  4038# 5788
SERIAL          0000   217# 6643
SETALLOCBIT     0D04  2717# 2800
SETARECORD      08C1  1949# 7052
SETARET         197F  3349  4047  4059  4730  5636  5642  5663  5700  5937#
SETBNKF         FF54   105# 3861  4200  4227  6325  7151
SETCDISK        0AC6  2330# 2503  4531  5370  5685
SETCDISK1       0AC9  2333# 2493
SETCDR          0B34  2414# 2900  3708  4824
SETDATA         0C52  2606# 3857  3999  4210  4223  5452  5980
SETDCNTDBLK     11D1  3158  3167  3656# 3679  5524
SETDCNTDBLK1    11D4  3658# 7501  7540
SETDIR1         0C58  2610# 4187
SETDLOG         0AC3  2328# 2473  4527
SETDMA          0C5B   695  2608  2613#
SETDMAF         FF24    89# 2003  7658# 7659
SETENDDIR       0C9D  2657# 2672  2816  2873  2980  3090  4622  5480  7491  7570
SETFCB          0A40  2163# 3854  3865  3870  4283  4289
SETFILEWF       13A1  4034# 4275  5932
SETFWF          0B20  2397# 2633  3475  3639  3715
SETHASH         1F14  2968  7376#
SETHASH1        1F22  7382  7384#
SETHASH15       1F3D  7392  7394#
SETHASH2        1F4A  7380  7399#
SETLRET1        06CF  1543# 1653# 3792  3801  3831  3834  3849  4335
SETLSN          0BFF  2536# 3782  4385  5586  5625  5705  5874  5913
SETMATCH        025D   527   534#  538
SETRANDOM       1655  4478# 6054
SETRC           10C4  2151  3491# 3556  3560  3805  4341
SETRC1          10D7  3497  3501#
SETRC2          10D9  3495  3503#
SETRC3          10DC  3506# 3626  6397  6399
SETRLOG         0BA6  2468  2500# 4525
SETRO           0B91  2491# 5995  5998
SETRR           1622  4452# 4469
SETSECF         FF21    88# 2002  7657# 7658
SETSIZE         1653  4462  4473#
SETSUBMITFLAG   100D  3335  3341#
```

```
SETTRKF         FF1E    87# 1985  7656# 7657
SETUSRCODE      19D5  6022  6025#
SETXDCNT        11CA  3653# 3749  4350  5806  6458  6542
SHELL           02CD   240   656#
SHELLERR        0335   668   697#
SHELLRTN        033E   663   701#
SHELLSP         0302   657   674#  704
SHELLSTK        030E   660   678#
SHL3BV          07C2  1771# 1928  2097  6215
SHL3BV1         07C3  1773# 1775
SHRPHYSHF       095B  1995  2005# 4004
SINGLE          1CC5  1839  2076  2782  2943  3570  4147  6429  6814#
SKPS            0299   408   469   581#  584   586
STAMP           1EF4   185# 4859  6590  6603  7333#
STAMP1          1774  4845# 5684  6463
STAMP2          1779  4847# 4921  5932  6466
STAMP3          177B  4846  4849#
STAMP4          1784  4852# 4870
STAMP5          179C  4868# 6463  6466
STARET          06CB   735  1465  1482  1489  1493  1497  1527  1530  1537# 1545
                      1557  1655# 2271  2319  2643  2911  3095  3786  4019  4135
                      4396  4410  5212  5229  5252  5419  5965  6024  6345  6496
                      6608
STHLRET         19C3   379  1550  5417  5961  5985  6004  6017# 6278  6652
STRTCOL         0716   970  1127  1191  1221  1239  1275  1292  1583#
SUBDH           0B3D  2312  2422# 2435  2670  6330  6990  7224  7506  7566
TAB             0009    45#  546   585   756   932
TABO            04B5   934#  944
TABOUT          04A0   632   740   914   924#  987  1459  1572
TABOUT1         04AF   926   930#
TBUFF           0080  1696# 5451
TEMPDRIVE       1EEC   181# 7327#
TEST152         0182   260   347#
TESTCTLSMODE    03A8   743   772#  777   788
TESTFFFF        0C97  2639  2649# 3679  5145  5146  5723  6997
TESTHASH        1FD2  7388  7462# 7467  7597
TESTMEDIAFLAG   0BCA  2515# 2836  4614
TESTPENDING     175F  4650  4656#
TESTVECTOR      0ADB  1892  2314  2344# 2502  2830  4492  4513
TESTVECTOR1     0ADE  2346# 5360  5361
TFCB            005C  1695#
TIMEF           FF4E   103# 4863  6593  6602
TMPSELECT       167C  4500# 5458  5464  6196  6242  6495
TOGGLEL1        0437   855   858#
TOGGLELISTCP    0425   849   853#
TRACK           1CFE  1984  6963# 6979  6990  6992  6997  7160
TRANV           1CBF  1829  1998  6786#
TRUE            00FF  1687# 1839  2879  3810  4414  6361
TRUNC1          1B46  6374# 6388
TRUNC15         1B61  6384# 6412
TRUNC2          1B64  6377  6386#
TRUNC3          1B6A  6381  6389#
TSTINVFCB       0C8A  2641# 3809  4064
TSTLOG0         0BBC  2508# 2522  2525
TSTLOG1         0BC0  2510# 2512
TSTLOGFXS       0BB3  2505# 2981  4605  7497  7536
```

```
TSTRELOG        0BE9  2528# 2696
TYPEAHEAD       1ECC   158#  782  7302#
UBYTES          000D  2021# 3027  4754
UPDATECS        0B8F  2448  2487#
UPDATESTAMP     17E8  4066  4910# 6372
USER0PASS       1CE2  2990  2992  3054  3101  6915#
USRCODE         1EE0   174# 1670# 4600  5505  6024  6026  7320#
UTILFLGS        1EA2   125# 7267#
VERSION         1EA1   121# 5419  7263#
VRECORD         1CCA  2044  2104  2149  2167  3786  3819  3825  3905  4069  4213
                      6828#
WBOOTF          FF03    67#   75#  652  7647# 7648
WILDMSG         076C   345  1647#
WORD            0002  1691# 6714  6720  6728  6729  6730  6731  6732  6734  6735
                      6736  6737  6738  6740  6760  6764  6765  6766  6767  6768
                      6775  6779  6786  6803  6837  6841  6865  6867  6868  6869
                      6870  6871  6872  6875  6876  6879  6899  6900  6906  6907
                      6908  6909  6951  6955  6956  6957  6958  6963  6964
WRBUFF          086A  1867# 4203  4239  7156
WRDIR           0C3E  2590# 3270  6371  6385
WRDIR0          0C46  2587  2595#
WRITE           14C8  4217  4222#
WRITE0          14DF  4236  4238#
WRITEF          FF2A    91# 1871  7660# 7661
XDCNT           1CDF   200# 3072  3138  3140  3154  3157  3164  3654  3657  3671
                      3679  4585  4704  5311  5506  5515  6900# 6906# 7544  7579
XDMAAD          1C8A   202# 2623  4755  4789  4820  4821  5218  5897  6218  6713#
XMOVEF          FF57   106# 7140
ZERO            0A62  2180# 2182  6504  6518  6523
ZERODM          1BD3  6391  6428#
ZERODM1         1BDC  6429  6431#
ZERODM2         1BE6  6435# 6437
ZEROEXTMOD      0A5A  2176# 6502  6537  6570
```

```
                          title   'CP/M BDOS Interface, BDOS, Version 3.0 Dec, 1982'
                  ;*****************************************************************
                  ;*****************************************************************
                  ;**                                                             **
                  ;**  B a s i c    D i s k   O p e r a t i n g   S y s t e m    **
                  ;**                                                             **
                  ;**              I n t e r f a c e   M o d u l e                **
                  ;**                                                             **
                  ;*****************************************************************
                  ;*****************************************************************
                  ;
                  ;       Copyright (c) 1978, 1979, 1980, 1981, 1982
                  ;       Digital Research
                  ;       Box 579, Pacific Grove
                  ;       California
                  ;
                  ;        December 1982
                  ;
 FFFF =           on      equ     0ffffh
 0000 =           off     equ     00000h
 0000 =           MPM     equ     off
 FFFF =           BANKED  equ     on

                  ;
                  ;       equates for non graphic characters
                  ;

 0001 =           ctla    equ     01h     ; control a
 0002 =           ctlb    equ     02h     ; control b
 0003 =           ctlc    equ     03h     ; control c
 0005 =           ctle    equ     05h     ; physical eol
 0006 =           ctlf    equ     06h     ; control f
 0007 =           ctlg    equ     07h     ; control g
 0008 =           ctlh    equ     08h     ; backspace
 000B =           ctlk    equ     0bh     ; control k
 0010 =           ctlp    equ     10h     ; prnt toggle
 0011 =           ctlq    equ     11h     ; start screen
 0012 =           ctlr    equ     12h     ; repeat line
 0013 =           ctls    equ     13h     ; stop screen
 0015 =           ctlu    equ     15h     ; line delete
 0017 =           ctlw    equ     17h     ; control w
 0018 =           ctlx    equ     18h     ; =ctl-u
 001A =           ctlz    equ     1ah     ; end of file
 007F =           rubout  equ     7fh     ; char delete
 0009 =           tab     equ     09h     ; tab char
 000D =           cr      equ     0dh     ; carriage return
 000A =           lf      equ     0ah     ; line feed
 005E =           ctl     equ     5eh     ; up arrow

 0000                             org     0000h
 0000 =           base            equ     $

                  ; Base page definitions

 FC00 =           bnkbdos$pg      equ     base+0fc00h
 FD00 =           resbdos$pg      equ     base+0fd00h
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
FB00 =          scb$pg          equ     base+0fb00h
FF00 =          bios$pg                 equ     base+0ff00h

                ; Bios equates

FF00 =          bios            equ     bios$pg
FF00 =          bootf           equ     bios$pg         ; 00. cold boot function

                if BANKED

FB68 =          wbootf          equ     scb$pg+68h      ; 01. warm boot function
FB6E =          constf          equ     scb$pg+6eh      ; 02. console status function
FB74 =          coninf          equ     scb$pg+74h      ; 03. console input function
FB7A =          conoutf                 equ     scb$pg+7ah      ; 04. console output function
FB80 =          listf           equ     scb$pg+80h      ; 05. list output function

                else

                wbootf          equ     bios$pg+3       ; 01. warm boot function
                constf          equ     bios$pg+6       ; 02. console status function
                coninf          equ     bios$pg+9       ; 03. console input function
                conoutf                 equ     bios$pg+12      ; 04. console output function
                listf           equ     bios$pg+15      ; 05. list output function

                endif

FF12 =          punchf          equ     bios$pg+18      ; 06. punch output function
FF15 =          readerf                 equ     bios$pg+21      ; 07. reader input function
FF18 =          homef           equ     bios$pg+24      ; 08. disk home function
FF1B =          seldskf                 equ     bios$pg+27      ; 09. select disk function
FF1E =          settrkf                 equ     bios$pg+30      ; 10. set track function
FF21 =          setsecf                 equ     bios$pg+33      ; 11. set sector function
FF24 =          setdmaf                 equ     bios$pg+36      ; 12. set dma function
FF27 =          readf           equ     bios$pg+39      ; 13. read disk function
FF2A =          writef          equ     bios$pg+42      ; 14. write disk function
FF2D =          liststf                 equ     bios$pg+45      ; 15. list status function
FF30 =          sectran                 equ     bios$pg+48      ; 16. sector translate
FF33 =          conoutstf       equ     bios$pg+51      ; 17. console output status function
FF36 =          auxinstf        equ     bios$pg+54      ; 18. aux input status function
FF39 =          auxoutstf       equ     bios$pg+57      ; 19. aux output status function
FF3C =          devtblf                 equ     bios$pg+60      ; 20. retunr device table address fx
FF3F =          devinitf        equ     bios$pg+63      ; 21. initialize device function
FF42 =          drvtblf                 equ     bios$pg+66      ; 22. return drive table address
FF45 =          multiof                 equ     bios$pg+69      ; 23. multiple i/o function
FF48 =          flushf          equ     bios$pg+72      ; 24. flush function
FF4B =          movef           equ     bios$pg+75      ; 25. memory move function
FF4E =          timef           equ     bios$pg+78      ; 26. system get/set time function
FF51 =          selmemf                 equ     bios$pg+81      ; 27. select memory function
FF54 =          setbnkf                 equ     bios$pg+84      ; 28. set dma bank function
FF57 =          xmovef          equ     bios$pg+87      ; 29. extended move function

                if BANKED

                ; System Control Block equates

FB90 =          olog            equ     scb$pg+090h
```

```
FB92 =          rlog            equ     scb$pg+092h

FB9C =          SCB             equ     scb$pg+09ch

                ; Expansion Area - 6 bytes

FB9C =          hashl           equ     scb$pg+09ch
FB9D =          hash            equ     scb$pg+09dh
FBA1 =          version                 equ     scb$pg+0a1h

                ; Utilities Section - 8 bytes

FBA2 =          util$flgs       equ     scb$pg+0a2h
FBA6 =          dspl$flgs       equ     scb$pg+0a6h

                ; CLP Section - 4 bytes

FBAA =          clp$flgs        equ     scb$pg+0aah
FBAC =          clp$errcde      equ     scb$pg+0ach

                ; CCP Section - 8 bytes

FBAE =          ccp$comlen      equ     scb$pg+0aeh
FBAF =          ccp$curdrv      equ     scb$pg+0afh
FBB0 =          ccp$curusr      equ     scb$pg+0b0h
FBB1 =          ccp$conbuff     equ     scb$pg+0b1h
FBB3 =          ccp$flgs        equ     scb$pg+0b3h

                ; Device I/O Section - 32 bytes

FBB6 =          conwidth        equ     scb$pg+0b6h
FBB7 =          column          equ     scb$pg+0b7h
FBB8 =          conpage                 equ     scb$pg+0b8h
FBB9 =          conline                 equ     scb$pg+0b9h
FBBA =          conbuffadd      equ     scb$pg+0bah
FBBC =          conbufflen      equ     scb$pg+0bch
FBBE =          conin$rflg      equ     scb$pg+0beh
FBC0 =          conout$rflg     equ     scb$pg+0c0h
FBC2 =          auxin$rflg      equ     scb$pg+0c2h
FBC4 =          auxout$rflg     equ     scb$pg+0c4h
FBC6 =          lstout$rflg     equ     scb$pg+0c6h
FBC8 =          page$mode       equ     scb$pg+0c8h
FBC9 =          pm$default      equ     scb$pg+0c9h
FBCA =          ctlh$act        equ     scb$pg+0cah
FBCB =          rubout$act      equ     scb$pg+0cbh
FBCC =          type$ahead      equ     scb$pg+0cch
FBCD =          contran                 equ     scb$pg+0cdh
FBCF =          conmode                 equ     scb$pg+0cfh
FBD3 =          outdelim        equ     scb$pg+0d3h
FBD4 =          listcp          equ     scb$pg+0d4h
FBD5 =          qflag           equ     scb$pg+0d5h

                ; BDOS Section - 42 bytes

FBD6 =          scbadd          equ     scb$pg+0d6h
FBD8 =          dmaad           equ     scb$pg+0d8h
```

```
FBDA =          olddsk          equ     scb$pg+0dah
FBDB =          info            equ     scb$pg+0dbh
FBDD =          resel           equ     scb$pg+0ddh
FBDE =          relog           equ     scb$pg+0deh
FBDF =          fx              equ     scb$pg+0dfh
FBE0 =          usrcode                 equ     scb$pg+0e0h
FBE1 =          dcnt            equ     scb$pg+0e1h
                ;searcha        equ     scb$pg+0e3h
FBE5 =          searchl                 equ     scb$pg+0e5h
FBE6 =          multcnt                 equ     scb$pg+0e6h
FBE7 =          errormode       equ     scb$pg+0e7h
FBE8 =          searchchain     equ     scb$pg+0e8h
FBEC =          temp$drive      equ     scb$pg+0ech
FBED =          errdrv          equ     scb$pg+0edh
FBF0 =          media$flag      equ     scb$pg+0f0h
FBF3 =          bdos$flags      equ     scb$pg+0f3h
FBF4 =          stamp           equ     scb$pg+0f4h
FBF9 =          commonbase      equ     scb$pg+0f9h
FBFB =          error           equ     scb$pg+0fbh     ;jmp error$sub
FBFE =          bdosadd                 equ     scb$pg+0feh

                ; Resbdos equates

FD00 =          resbdos                 equ     resbdos$pg
FD09 =          move$out        equ     resbdos$pg+9    ; a=bank #, hl=dest, de=srce
FD0C =          move$tpa        equ     resbdos$pg+0ch  ; a=bank #, hl=dest, de=srce
FD0F =          srch$hash       equ     resbdos$pg+0fh  ; a=bank #, hl=hash table addr
FD12 =          hashmx          equ     resbdos$pg+12h  ; max hash search dcnt
FD14 =          rd$dir$flag     equ     resbdos$pg+14h  ; directory read flag
FD15 =          make$xfcb       equ     resbdos$pg+15h  ; make function flag
FD16 =          find$xfcb       equ     resbdos$pg+16h  ; search function flag
FD17 =          xdcnt           equ     resbdos$pg+17h  ; dcnt save for empty fcb,
                                                        ; user 0 fcb, or xfcb
FD19 =          xdmaad          equ     resbdos$pg+19h  ; resbdos dma copy area addr
FD1B =          curdma          equ     resbdos$pg+1bh  ; current dma
FD1D =          copy$cr$only    equ     resbdos$pg+1dh  ; dont restore fcb flag
FD1E =          user$info       equ     resbdos$pg+1eh  ; user fcb address
FD20 =          kbchar          equ     resbdos$pg+20h  ; conbdos look ahead char
FD21 =          qconinx                 equ     resbdos$pg+21h  ; qconin mov a,m routine

                ELSE

                move$out        equ     movef
                move$tpa        equ     movef

                ENDIF

                ;
0000 3635343332serial: db       '654321'
                ;
                ;       Enter here from the user's program with function number in c,
                ;       and information address in d,e
                ;

                bdose: ; Arrive here from user programs
0006 EB22DBFBEB         xchg! shld info! xchg ; info=de, de=info
```

```

0008 7932DFFBFE         mov a,c! sta fx! cpi 14! jc bdose2
0014 2100002298         lxi h,0! shld dircnt ; dircnt,multnum = 0
001A 3ADAFB32A4         lda olddsk! sta seldsk ; Set seldsk

                if BANKED
0020 3D32BC28           dcr a! sta copy$cr$init
                ENDIF

                        ; If mult$cnt ~= 1 then read or write commands
                        ; are handled by the shell
0024 3AE6FB3DCA         lda mult$cnt! dcr a! jz bdose2
002B 219500             lxi h,mult$fxs
                bdose1:
002E 7EB7CA3B00         mov a,m! ora a! jz bdose2
0033 B9CA3403           cmp c! jz shell
0037 23C32E00           inx h! jmp bdose1
                bdose2:
003B 7B329E28           mov a,e! sta linfo ; linfo = low(info) - don't equ
003F 210000229F         lxi h,0! shld aret ; Return value defaults to 0000
0045 22DDFB             shld resel ; resel,relog = 0
                        ; Save user's stack pointer, set to local stack
0048 39223203           dad sp! shld entsp ; entsp = stackptr

                if not BANKED
                        lxi sp,lstack ; local stack setup
                ENDIF

004C 213B28             lxi h,goback ; Return here after all functions
004F E5                 push h ; jmp goback equivalent to ret
0050 79FE33D25D         mov a,c! cpi nfuncs! jnc high$fxs ; Skip if invalid #
0056 4B                 mov c,e ; possible output character to c
0057 21C600C36F         lxi h,functab! jmp bdos$jmp
                        ; look for functions 98 ->
                high$fxs:
005D FE80D25C01         cpi 128! jnc test$152
0062 D662DAC015         sui 98! jc lret$eq$ff ; Skip if function < 98
0067 FE0FD2C015         cpi nfuncs2! jnc lret$eq$ff
006C 212C01             lxi h,functab2
                bdos$jmp:
006F 5F1600             mov e,a! mvi d,0 ; de=func, hl=.ciotab
0072 19195E2356         dad d! dad d! mov e,m! inx h! mov d,m ; de=functab(func)
0077 2ADBFB             lhld info ; info in de for later xchg
007A EBE9               xchg! pchl ; dispatched

                ;          CAUTION: In banked systems only,
                ;           error$sub is referenced indirectly by the SCB ERROR
                ;          field in RESBDOS as (0fc7ch).  This value is converted
                ;          to the actual address of error$sub by GENSYS.  If the offset
                ;          of error$sub is changed, the SCB ERROR value must also
                ;          be changed.


                ;
                ;       error subroutine
                ;

```

```
                        error$sub:
     007C 0600C50D              mvi b,0! push b! dcr c
     0080 214A010909            lxi h,errtbl! dad b! dad b
     0085 5E2356EB              mov e,m! inx h! mov d,m! xchg
     0089 CD8002                call errflg
     008C C13AE7FBB7            pop b! lda error$mode! ora a! rnz
     0092 C32303                jmp reboote

     0095 1415212228mult$fxs:        db      20,21,33,34,40,0

                        if BANKED
     009B 434F505952            db      'COPYRIGHT (C) 1982,'
     00AE 2044494749            db      ' DIGITAL RESEARCH '
     00C0 3135313238            db      '151282'
                        else
                                db      'COPR. ''82 DRI 151282'

                                ;       31 level stack

                                dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
                                dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
                                dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
                                dw      0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h
                        lstack:

                        endif

                        ;       dispatch table for functions

                        functab:
     00C6 2F03C20328            dw      rebootx1, func1, func2, func3
     00CE 12FF80FB07            dw      punchf, listf, func6, func7
     00D6 310A370AC8            dw      func8, func9, func10, func11
     000C =             diskf   equ     ($-functab)/2   ; disk funcs
     00DE 7D21832196            dw      func12,func13,func14,func15
     00E6 87229B22EC            dw      func16,func17,func18,func19
     00EE FB2204230D            dw      func20,func21,func22,func23
     00F6 F523FB2301            dw      func24,func25,func26,func27
     00FE D311112417            dw      func28,func29,func30,func31
     0106 2D2441244A            dw      func32,func33,func34,func35
     010E 401E592475            dw      func36,func37,func38,func39
     0116 4A24C01575            dw      func40,lret$eq$ff,func42,func43
     011E 7D248B2490            dw      func44,func45,func46,func47
     0126 DF242A2550            dw      func48,func49,func50
     0033 =             nfuncs equ      ($-functab)/2

                        functab2:
     012C E824C825              dw      func98,func99
     0130 A6261F2728            dw      func100,func101,func102,func103
     0138 F127FE2713            dw      func104,func105,func106,func107
     0140 27287B0A89            dw      func108,func109,func110,func111
     0148 950A                  dw      func112

     000F =             nfuncs2         equ     ($-functab2)/2

                        errtbl:
```

```
014A 410D               dw      permsg
014C 670D               dw      rodmsg
014E 580D               dw      rofmsg
0150 4A0D               dw      selmsg
0152 0000               dw      0
0154 0000               dw      0
0156 760D               dw      passmsg
0158 850D               dw      fxstsmsg
015A 910D               dw      wildmsg

                test$152:
015C FE98C0             cpi 152! rnz

                ;
                ;       PARSE version 3.0b  Oct 08 1982 - Doug Huskey
                ;
                ;
                        ; DE->.(.filename,.fcb)
                        ;
                        ; filename = [d:]file[.type][;password]
                        ;
                        ; fcb assignments
                        ;
                        ;   0     => drive, 0 = default, 1 = A, 2 = B, ...
                        ;   1-8   => file, converted to upper case,
                        ;            padded with blanks (left justified)
                        ;   9-11  => type, converted to upper case,
                        ;            padded with blanks (left justified)
                        ;   12-15 => set to zero
                        ;   16-23 => password, converted to upper case,
                        ;            padded with blanks
                        ;   24-25 => 0000h
                        ;   26    => length of password (0 - 8)
                        ;
                        ; Upon return, HL is set to FFFFH if DE locates
                        ;            an invalid file name;
                        ; otherwise, HL is set to 0000H if the delimiter
                        ;            following the file name is a 00H (NULL)
                        ;            or a 0DH (CR);
                        ; otherwise, HL is set to the address of the delimiter
                        ;            following the file name.
                        ;
015F 212924             lxi     h,sthl$ret
0162 E5                 push    h
0163 2ADBFB             lhld    info
0166 5E                 mov     e,m             ;get first parameter
0167 23                 inx     h
0168 56                 mov     d,m
0169 D5                 push    d               ;save .filename
016A 23                 inx     h
016B 5E                 mov     e,m             ;get second parameter
016C 23                 inx     h
016D 56                 mov     d,m
016E E1                 pop     h               ;DE=.fcb  HL=.filename
016F EB                 xchg
                parse0:
```

```
0170 E5                 push    h               ;save .fcb
0171 AF                 xra     a
0172 77                 mov     m,a             ;clear drive byte
0173 23                 inx     h
0174 010B20             lxi     b,20h*256+11
0177 CD6C02             call    pad             ;pad name and type w/ blanks
017A 010400             lxi     b,4
017D CD6C02             call    pad             ;EXT, S1, S2, RC = 0
0180 010820             lxi     b,20h*256+8
0183 CD6C02             call    pad             ;pad password field w/ blanks
0186 010C00             lxi     b,12
0189 CD6C02             call    pad             ;zero 2nd 1/2 of map, cr, r0 - r2
                ;
                ;       skip spaces
                ;
018C CD7302             call    skps
                ;
                ;       check for drive
                ;
018F 1A                 ldax    d
0190 FE3A               cpi     ':'             ;is this a drive?
0192 1B                 dcx     d
0193 E1                 pop     h
0194 E5                 push    h               ;HL = .fcb
0195 C2AC01             jnz     parse$name
                ;
                ;       Parse the drive-spec
                ;
                parsedrv:
0198 CD4D02             call    delim
019B CAE001             jz      parse$ok
019E D641               sui     'A'
01A0 DAF601             jc      perror1
01A3 FE10               cpi     16
01A5 D2F601             jnc     perror1
01A8 13                 inx     d
01A9 13                 inx     d               ;past the ':'
01AA 3C                 inr     a               ;set drive relative to 1
01AB 77                 mov     m,a             ;store the drive in FCB(0)
                ;
                ;       Parse the file-name
                ;
                parse$name:
01AC 23                 inx     h               ;HL = .fcb(1)
01AD CD4D02             call    delim
01B0 CAE001             jz      parse$ok
01B3 010007             lxi     b,7*256

01B6 1A         parse_;         ldax    d               ;get a character
01B7 FE2E               cpi     '.'             ;file-type next?
01B9 CACA01             jz      parse$type      ;branch to file-type processing
01BC FE3B               cpi     ';'
01BE CAFB01             jz      parse$pw
01C1 CD1902             call    gfc             ;process one character
01C4 C2B601             jnz     parse6          ;loop if not end of name
01C7 C3E001             jmp     parse$ok
```

```
                    ;
                    ;       Parse the file-type
                    ;
                    parse$type:
01CA 13                     inx     d               ;advance past dot
01CB E1                     pop     h
01CC E5                     push    h               ;HL =.fcb
01CD 010900                 lxi     b,9
01D0 09                     dad     b               ;HL =.fcb(9)
01D1 010002                 lxi     b,2*256

01D4 1A             parse8:         ldax    d
01D5 FE3B                   cpi     ';'
01D7 CAFB01                 jz      parsepw
01DA CD1902                 call    gfc             ;process one character
01DD C2D401                 jnz     parse8          ;loop if not end of type
                    ;
                    parse$ok:
01E0 C1                     pop     b
01E1 D5                     push    d
01E2 CD7302                 call    skps            ;skip trailing blanks and tabs
01E5 1B                     dcx     d
01E6 CD4D02                 call    delim           ;is next nonblank char a delim?
01E9 E1                     pop     h
01EA C0                     rnz                     ;no
01EB 210000                 lxi     h,0
01EE B7                     ora     a
01EF C8                     rz                      ;return zero if delim = 0
01F0 FE0D                   cpi     cr
01F2 C8                     rz                      ;return zero if delim = cr
01F3 EB                     xchg
01F4 C9                     ret
                    ;
                    ;       handle parser error
                    ;
                    perror:
01F5 C1                     pop     b                       ;throw away return addr
                    perror1:
01F6 C1                     pop     b
01F7 21FFFF                 lxi     h,0ffffh
01FA C9                     ret
                    ;
                    ;       Parse the password
                    ;
                    parsepw:
01FB 13                     inx     d
01FC E1                     pop     h
01FD E5                     push    h
01FE 011000                 lxi     b,16
0201 09                     dad     b
0202 010107                 lxi     b,7*256+1
                    parsepw1:
0205 CD1902                 call    gfc
0208 C20502                 jnz     parsepw1
020B 3E07                   mvi     a,7
020D 90                     sub     b
```

```
 020E E1               pop     h
 020F E5               push    h
 0210 011A00           lxi     b,26
 0213 09               dad     b
 0214 77               mov     m,a
 0215 1A               ldax    d                       ;delimiter in A
 0216 C3E001           jmp     parse$ok
                ;
                ;       get next character of name, type or password
                ;
 0219 CD4D02    gfc:   call    delim           ;check for end of filename
 021C C8               rz                      ;return if so
 021D FE20             cpi     ' '             ;check for control characters
 021F 13               inx     d
 0220 DAF501           jc      perror          ;error if control characters encountered
 0223 04               inr     b               ;error if too big for field
 0224 05               dcr     b
 0225 FAF501           jm      perror
 0228 0C               inr     c
 0229 0D               dcr     c
 022A C23202           jnz     gfc1
 022D FE2A             cpi     '*'             ;trap 'match rest of field' character
 022F CA3702           jz      setmatch
 0232 77        gfc1:  mov     m,a             ;put character in fcb
 0233 23               inx     h
 0234 05               dcr     b               ;decrement field size counter
 0235 B7               ora     a               ;clear zero flag
 0236 C9               ret
                ;;
                setmatch:
 0237 363F             mvi     m,'?'           ;set match one character
 0239 23               inx     h
 023A 05               dcr     b
 023B F23702           jp      setmatch
 023E C9               ret
                ;
                ;       check for delimiter
                ;
                ;       entry:  A = character
                ;       exit:   z = set if char is a delimiter
                ;
 023F 0D09202E2Cdelimiters:    db      cr,tab,' .,:;[]=<>|',0

 024D 1A        delim: ldax    d               ;get character
 024E E5               push    h
 024F 213F02           lxi     h,delimiters
 0252 BE        delim1:        cmp     m               ;is char in table
 0253 CA5D02           jz      delim2
 0256 34               inr     m
 0257 35               dcr     m               ;end of table? (0)
 0258 23               inx     h
 0259 C25202           jnz     delim1
 025C B7               ora     a               ;reset zero flag
 025D E1        delim2:        pop     h
 025E C8               rz
                       ;
```

```
                          ;       not a delimiter, convert to upper case
                          ;
025F FE61                 cpi     'a'
0261 D8                   rc
0262 FE7B                 cpi     'z'+1
0264 D26902               jnc     delim3
0267 E65F                 ani     05fh
0269 E67F         delim3:         ani     07fh
026B C9                   ret                             ;return with zero set if so
                  ;
                  ;       pad with blanks or zeros
                  ;
026C 70           pad:    mov     m,b
026D 23                   inx     h
026E 0D                   dcr     c
026F C26C02               jnz     pad
0272 C9                   ret
                  ;
                  ;       skip blanks and tabs
                  ;
0273 1A           skps:   ldax    d
0274 13                   inx     d
0275 FE20                 cpi     ' '                     ;skip spaces & tabs
0277 CA7302               jz      skps
027A FE09                 cpi     tab
027C CA7302               jz      skps
027F C9                   ret
                  ;
                  ;       end of PARSE
                  ;

                  errflg:
                          ; report error to console, message address in hl
0280 E5CD8205             push h! call crlf ; stack mssg address, new line
0284 3AAA28C641           lda adrive! adi 'A'! sta dskerr ; current disk name
028C 012F0D               lxi b,dskmsg

                  if BANKED
028F CDEE02               call zprint ; the error message
                  else
                          call print
                  endif

0292 C1                   pop b

                  if BANKED
0293 3AF3FB17D2           lda bdos$flags! ral! jnc zprint
029A CDEE02               call zprint ; error message tail
029D 3ADFFB0630           lda fx! mvi b,30h
02A2 210B03               lxi h,pr$fx1
02A5 FE64DAAF02           cpi 100! jc errflg1
02AA 363123D664           mvi m,31h! inx h! sui 100
                  errflg1:
02AF D60ADAB802           sui 10! jc errflg2
02B4 04C3AF02             inr b! jmp errflg1
                  errflg2:
```

```
02B8 7023C63A77       mov m,b! inx h! adi 3ah! mov m,a
02BD 233620           inx h! mvi m,20h
02C0 210E033600       lxi h,pr$fcb! mvi m,0
02C5 3ADDFBB7CA       lda resel! ora a! jz errflg3
02CC 3620D5           mvi m,20h! push d
02CF 2ADBFB23EB       lhld info! inx h! xchg! lxi h,pr$fcb1
02D7 0E08CD290E       mvi c,8! call move! mvi m,'.'! inx h
02DF 0E03CD290E       mvi c,3! call move! pop d
               errflg3:
02E5 CD8205           call crlf
02E8 01FB02C3EE       lxi b,pr$fx! jmp zprint

               zprint:
02EE 0AB7C8           ldax b! ora a! rz
02F1 C54F             push b! mov c,a
02F3 CD2805           call tabout
02F6 C103C3EE02       pop b! inx b! jmp zprint

02FB 42444F5320pr$fx: db      'BDOS Function = '
030B 202020    pr$fx1:        db      '   '
030E 2046696C65pr$fcb:        db      ' File = '
0316           pr$fcb1:ds     12
0322 00               db      0

               else
                      jmp     print
               endif

               reboote:
0323 21FDFFC32C       lxi h,0fffdh! jmp rebootx0 ; BDOS error
               rebootx:
0329 21FEFF           lxi h,0fffeh ; CTL-C error
               rebootx0:
032C 22ACFB           shld clp$errcde
               rebootx1:
032F C368FB           jmp wbootf

0332           entsp: ds      2       ; entry stack pointer

               shell:
0334 2100003922       lxi h,0! dad sp! shld shell$sp

               if not BANKED
                      lxi sp,shell$stk
               endif

033B 21A203E5         lxi h,shell$rtn! push h
033F CD090ECD89       call save$rr! call save$dma
0345 3AE6FB           lda mult$cnt
               mult$io:
0348 F5329928CD       push a! sta mult$num! call cbdos
034F B7C29903         ora a! jnz shell$err
0353 3ADFFBFE21       lda fx! cpi 33! cnc incr$rr
035B CD7F03           call adv$dma
035E F13DC24803       pop a! dcr a! jnz mult$io
0363 676FC9           mov h,a! mov l,a! ret
```

```

0366 0000    shell$sp:     dw     0

0368 C7C7C7C7C7              dw     0c7c7h,0c7c7h,0c7c7h,0c7c7h,0c7c7h

             shell$stk:           ; shell has 5 level stack
0372 0000    hold$dma:     dw     0

             cbdos:
0374 3ADFFB4F      lda fx! mov c,a
             cbdos1:
0378 2ADBFBEBC3    lhld info! xchg! jmp bdose2

             adv$dma:
037F 2AD8FB1180    lhld dmaad! lxi d,80h! dad d! jmp reset$dma1

             save$dma:
0389 2AD8FB2272    lhld dmaad! shld hold$dma! ret

             reset$dma:
0390 2A7203        lhld hold$dma
             reset$dma1:
0393 22D8FBC3BF    shld dmaad! jmp setdma

             shell$err:
0399 C13CC8        pop b! inr a! rz
039C 3AE6FB9067    lda mult$cnt! sub b! mov h,a! ret

             shell$rtn:
03A2 E53ADFFBFE    push h! lda fx! cpi 33! cnc reset$rr
03AB CD9003        call reset$dma
03AE D12A6603F9    pop d! lhld shell$sp! sphl! xchg
03B4 7D44C9        mov a,l! mov b,h! ret

                   page
```

```



                       title   'CP/M Bdos Interface, Bdos, Version 3.0 Nov, 1982'
                ;**************************************************************
                ;**************************************************************
                ;**                                                          **
                ;**  B a s i c   D i s k   O p e r a t i n g   S y s t e m   **
                ;**                                                         **
                ;**              C o n s o l e   P o r t i o n               **
                ;**                                                          **
                ;**************************************************************
                ;**************************************************************
                ;
                ;       November 1982
                ;
                ;
                ;      Console handlers
                ;
                conin:
                       ;read console character to A
03B7 2120FD7E36        lxi h,Kbchar! mov a,m! mvi m,0! ora a! rnz
                       ;no previous Keyboard character ready
03BF C374FB            jmp coninf ;get character externally
                       ;ret
                ;
                conech:
03C2 21720AE5          LXI H,STA$RET! PUSH H
                CONECHO:
                       ;read character with echo
03C6 CDB703CDF0        call conin! call echoc! JC CONECH1 ;echo character?
                        ;character must be echoed before return
03CF F54FCD2805        push psw! mov c,a! call tabout! pop psw
03D5 C9                RET
                CONECH1:
03D6 CD1304C0          CALL TEST$CTLS$MODE! RNZ
03DA FE13C2E503        CPI CTLS! JNZ CONECH2
03DF CD5E04C3C6        CALL CONBRK2! JMP CONECHO
                CONECH2:
03E5 FE11CAC603        CPI CTLQ! JZ CONECHO
03EA FE10CAC603        CPI CTLP! JZ CONECHO
03EF C9                RET
                ;
                echoc:
                       ;echo character if graphic
                       ;cr, lf, tab, or backspace
03F0 FE0DC8            cpi cr! rz ;carriage return?
03F3 FE0AC8            cpi lf! rz ;line feed?
03F6 FE09C8            cpi tab! rz ;tab?
03F9 FE08C8            cpi ctlh! rz ;backspace?
03FC FE20C9            cpi ' '! ret ;carry set if not graphic
                ;
                CONSTX:
03FF 3A20FDB7C2        LDA KBCHAR! ORA A! JNZ CONB1
0406 CD6EFBE601        CALL CONSTF! ANI 1! RET
                ;
```

```
               if BANKED

               SET$CTLS$MODE:
                      ;SET CTLS STATUS OR INPUT FLAG FOR QUEUE MANAGER
040C 21D5FB3640       LXI H,QFLAG! MVI M,40H! XTHL! PCHL

               endif
               ;
               TEST$CTLS$MODE:
                      ;RETURN WITH Z FLAG RESET IF CTL-S CTL-Q CHECKING DISABLED
0413 473ACFFBE6       MOV B,A! LDA CONMODE! ANI 2! MOV A,B! RET
               ;
               conbrk:        ;check for character ready
041B CD1304C2FF       CALL TEST$CTLS$MODE! JNZ CONSTX
0421 3A20FDB7C2       lda kbchar! ora a! jnz CONBRK1 ;skip if active kbchar
                              ;no active kbchar, check external break
                              ;DOES BIOS HAVE TYPE AHEAD?
               if BANKED
0428 3ACCFB3CCA               LDA TYPE$AHEAD! INR A! JZ CONSTX ;YES
               endif
                              ;CONBRKX CALLED BY CONOUT

                      CONBRKX:
                              ;HAS CTL-S INTERCEPT BEEN DISABLED?
042F CD1304C0                 CALL TEST$CTLS$MODE! RNZ ;YES
                              ;DOES KBCHAR CONTAIN CTL-S?
0433 3A20FDFE13               LDA KBCHAR! CPI CTLS! JZ CONBRK1 ;YES
               if BANKED
043B CD0C04                   CALL SET$CTLS$MODE
               endif
                              ;IS A CHARACTER READY FOR INPUT?
043E CD6EFB                   call constf
               if BANKED
0441 E13600                   POP H! MVI M,0
               endif
0444 E601C8                   ani 1! rz ;NO
                              ;character ready, read it
               if BANKED
0447 CD0C04                   CALL SET$CTLS$MODE
               endif
044A CD74FB                   call coninf
               if BANKED
044D E13600                   POP H! MVI M,0
               endif
                      CONBRK1:
0450 FE13C27F04               cpi ctls! jnz conb0 ;check stop screen function
                              ;DOES KBCHAR CONTAIN A CTL-S?
0455 2120FDBEC2               LXI H,KBCHAR! CMP M! JNZ CONBRK2 ;NO
045C 3600                     MVI M,0 ; KBCHAR = 0
                              ;found ctls, read next character
                      CONBRK2:

               if BANKED
045E CD0C04                   CALL SET$CTLS$MODE
               endif
0461 CD74FB                   call coninf ;to A
```

```
                if BANKED
0464 E13600                     POP H! MVI M,0
                endif
0467 FE03C27504                 cpi ctlc! JNZ CONBRK3
046C 3ACFFBE608                 LDA CONMODE! ANI 08H! JZ REBOOTX
0474 AF                         XRA A
                        CONBRK3:
0475 D611C8                     SUI CTLQ! RZ ; RETURN WITH A = ZERO IF CTLQ
0478 3CCDA004C3                 INR A! CALL CONB3! JMP CONBRK2
                        conb0:
047F 2120FD                     LXI H,KBCHAR

0482 47                         MOV B,A
                                ;IS CONMODE(1) TRUE?
0483 3ACFFB1FD2                 LDA CONMODE! RAR! JNC $+7 ;NO
                                ;DOES KBCHAR = CTLC?
048A 3E03BEC8                   MVI A,CTLC! CMP M! RZ ;YES - RETURN
048E 78                         MOV A,B

048F FE11CA9D04                 CPI CTLQ! JZ CONB2
0494 FE10CA9D04                 CPI CTLP! JZ CONB2
                                ;character in accum, save it
0499 77                         MOV M,A
                        conb1:
                                ;return with true set in accumulator
049A 3E01C9                     mvi a,1! ret
                        CONB2:
049D AF77C9                     XRA A! MOV M,A! RET
                        CONB3:
04A0 CCA904                     CZ TOGGLE$LISTCP
04A3 0E07C47AFB                 MVI C,7! CNZ CONOUTF
04A8 C9                         RET
                ;
                TOGGLE$LISTCP:
                        ; IS PRINTER ECHO DISABLED?
04A9 3ACFFBE614         LDA CONMODE! ANI 14H! JNZ TOGGLE$L1 ;YES
04B1 21D4FB3E01         LXI H,LISTCP! MVI A,1! XRA M! ANI 1
04B9 77C9               MOV M,A! RET
                TOGGLE$L1:
04BB AFC9               XRA A! RET
                ;
                QCONOUTF:
                        ;DOES FX = INPUT?
04BD 3ADFFB3DCA         LDA FX! DCR A! JZ CONOUTF ;YES
                        ;IS ESCAPE SEQUENCE DECODING IN EFFECT?
04C4 78E608C27A         MOV A,B! ANI 8! JNZ SCONOUTF ;YES
04CA C37AFB             JMP CONOUTF
                ;
                conout:
                        ;compute character position/write console char from C
                        ;compcol = true if computing column position
04CD 3ABC0AB7C2         lda compcol! ora a! jnz compout
                                ;write the character, then compute the column
                                ;write console character from C
                                ;B ^= 0 -> ESCAPE SEQUENCE DECODING
04D4 3ACFFBE614                 LDA CONMODE! ANI 14H! MOV B,A
```

```
04DA C5                         push b
                                ;CALL CONBRKX FOR OUTPUT FUNCTIONS ONLY
04DB 3ADFFB3DC4                 LDA FX! DCR A! CNZ CONBRKX
04E2 C1C5                       pop b! push b ;recall/save character
04E4 CDBD04                     call QCONOUTF ;externally, to console
04E7 C1                         pop b
                                ;SKIP ECHO WHEN CONMODE & 14H ^= 0
04E8 78B7C2F604                 MOV A,B! ORA A! JNZ COMPOUT
04ED C5                         push b ;recall/save character
                                ;may be copying to the list device
04EE 3AD4FBB7C4                 lda listcp! ora a! cnz listf ;to printer, if so
04F5 C1                         pop b ;recall the character
                        compout:
04F6 79                         mov a,c ;recall the character
                                ;and compute column position
04F7 21B7FB                     lxi h,column ;A = char, HL = .column
04FA FE7FC8                     cpi rubout! rz ;no column change if nulls
04FD 34                         inr m ;column = column + 1
04FE FE20D0                     cpi ' '! rnc ;return if graphic
                                ;not graphic, reset column position
0501 35                         dcr m ;column = column - 1
0502 7EB7C8                     mov a,m! ora a! rz ;return if at zero
                                ;not at zero, may be backspace or end line
0505 79                         mov a,c ;character back to A
0506 FE08C20D05                 cpi ctlh! jnz notbacksp
                                        ;backspace character
050B 35                                 dcr m ;column = column - 1
050C C9                                 ret
                                notbacksp:
                                        ;not a backspace character, eol?
050D FE0DC0                             cpi cr! rnz ;return if not
                                        ;end of line, column = 0
0510 3600                               mvi m,0 ;column = 0
0512 C9                         ret
                ;
                ctlout:
                        ;send C character with possible preceding up-arrow
0513 79CDF003           mov a,c! call echoc ;cy if not graphic (or special case)
0517 D22B05             jnc tabout ;skip if graphic, tab, cr, lf, or ctlh
                                ;send preceding up arrow
051A F50E5ECDCD                 push psw! mvi c,ctl! call conout ;up arrow
0520 F1F640                     pop psw! ori 40h ;becomes graphic letter
0523 4F                         mov c,a ;ready to print
                if BANKED
0524 CDC005C8                   call chk$column! rz
                endif
                                ;(drop through to tabout)
                ;
                tabout:
                        ;IS FX AN INPUT FUNCTION?
052B 3ADFFB3DCA         LDA FX! DCR A! JZ TABOUT1 ;YES - ALWAYS EXPAND TABS FOR ECHO
                        ;HAS TAB EXPANSION BEEN DISABLED OR
                        ;ESCAPE SEQUENCE DECODING BEEN ENABLED?
052F 3ACFFBE614         LDA CONMODE! ANI 14H! JNZ CONOUT ;YES
                TABOUT1:
                        ;expand tabs to console
```

```
0537 79FE09C2CD        mov a,c! cpi tab! jnz conout ;direct to conout if not
                               ;tab encountered, move to next tab position
                       tab0:

                if BANKED
053D 3ADFFBFE01                lda fx! cpi 1! jnz tab1
0545 CDC005C8                  call chk$column! rz
                       tab1:
                endif

0549 0E20CDCD04                mvi c,' '! call conout ;another blank
054E 3AB7FBE607                lda column! ani 111b ;column mod 8 = 0 ?
0553 C23D05                    jnz tab0 ;back for another if not
0556 C9                ret
                ;
                ;
                backup:
                       ;back-up one screen position
0557 CD6505            call pctlh

                if BANKED
055A 3ABF0AFE01        lda comchr! cpi ctla! rz
                endif

0560 0E20CD7AFB        mvi c,' '! call conoutf
                ;      (drop through to pctlh)                         ;
                pctlh:
                       ;send ctlh to console without affecting column count
0565 0E08C37AFB        mvi c,ctlh! jmp conoutf
                       ;ret
                ;
                crlfp:
                       ;print #, cr, lf for ctlx, ctlu, ctlr functions
                       ;then move to strtcol (starting column)
056A 0E23CDCD04        mvi c,'#'! call conout
056F CD8205            call crlf
                       ;column = 0, move to position strtcol
                       crlfp0:
0572 3AB7FB21BD                lda column! lxi h,strtcol
0578 BED0                      cmp m! rnc ;stop when column reaches strtcol
057A 0E20CDCD04                mvi c,' '! call conout ;print blank
057F C37205                    jmp crlfp0
                ;;
                ;
                crlf:
                       ;carriage return line feed sequence
0582 0E0DCDCD04        mvi c,cr! call conout! mvi c,lf! jmp conout
                       ;ret
                ;
                print:
                       ;print message until M(BC) = '$'
058C 21D3FB            LXI H,OUTDELIM
058F 0ABEC8            ldax b! CMP M! rz ;stop on $
                               ;more to print
0592 03C54F                    inx b! push b! mov c,a ;char to C
0595 CD2805                    call tabout ;another character printed
```

```
0598 C1C38C05                           pop b! jmp print
                    ;
                    QCONIN:

                    if BANKED
059C 2A280D7E32             lhld apos! mov a,m! sta ctla$sw
                    endif
                            ;IS BUFFER ADDRESS = 0?
05A3 2ABAFB7DB4             LHLD CONBUFFADD! MOV A,L! ORA H! JZ CONIN ;YES
                            ;IS CHARACTER IN BUFFER < 5?

                    if BANKED
05A8 CD21FD                 call qconinx ; mov a,m with bank 1 switched in
                    else
                            MOV A,M
                    endif

05AE 23                     INX H
05AF B7C2B605               ORA A! JNZ QCONIN1 ; NO
05B3 210000                 LXI H,0
                    QCONIN1:
05B6 22BAFB22BC             SHLD CONBUFFADD! SHLD CONBUFFLEN! RNZ ; NO
05BD C3B703                 JMP CONIN

                    if BANKED

                    chk$column:
05C0 3AB6FB5F3A             lda conwidth! mov e,a! lda column! cmp e! ret
                    ;
                    expand:
05C9 EB2A280DEB             xchg! lhld apos! xchg
                    expand1:
05CE 1AB7C8                 ldax d! ora a! rz
05D1 13237704C3             inx d! inx h! mov m,a! inr b! jmp expand1
                    ;
                    copy$xbuff:
05D8 78B7C8                 mov a,b! ora a! rz
05DB C548E5EB13             push b! mov c,b! push h! xchg! inx d
05E0 21C20B                 lxi h,xbuff
05E3 CD290E                 call move
05E6 360022260D             mvi m,0! shld xpos
05EB E1C1C9                 pop h! pop b! ret
                    ;
                    copy$cbuff:
05EE 3AB4FB17D0             lda ccpflgs+1! ral! rnc
05F3 21C20B11C1             lxi h,xbuff! lxi d,cbuff! inr c! jnz copy$cbuff1
05FD EB78B7C8               xchg! mov a,b! ora a! rz
0601 32C00A                 sta cbuff$len
0604 D5010F06C5             push d! lxi b,copy$cbuff2! push b
0609 47                     mov b,a
                    copy$cbuff1:
060A 0448C3290E             inr b! mov c,b! jmp move
                    copy$cbuff2:
060F E12B3600C9             pop h! dcx h! mvi m,0! ret
                    ;
                    save$col:
```

```
0614 3AB7FB322C         lda column! sta save$column! ret
                  ;
                  clear$right:
061B 3AB7FB212D         lda column! lxi h,ctla$column! cmp m! rnc
0623 0E20CDCD04         mvi c,20h! call conout! jmp clear$right
                  ;
                  reverse:
062B 3A2C0D21B7         lda save$column! lxi h,column! cmp m! rnc
0633 0E08CDCD04         mvi c,ctlh! call conout! jmp reverse
                  ;
                  chk$buffer$size:
063B C5E5               push b! push h
063D 2A280D1E00         lhld apos! mvi e,0
                  cbs1:
0642 7EB7CA4C06         mov a,m! ora a! jz cbs2
0647 1C23C34206         inr e! inx h! jmp cbs1
                  cbs2:
064C 7883B9             mov a,b! add e! cmp c
064F F50E07D47A         push a! mvi c,7! cnc conoutf
0655 F1E1C1D8           pop a! pop h! pop b! rc
0659 D1D1C3FF06         pop d! pop d! jmp readnx
                  ;
                  refresh:
065E 3A2A0DB7C8         lda ctla$sw! ora a! rz
0663 3ABF0AFE01         lda comchr! cpi ctla! rz
0669 FE06C8             cpi ctlf! rz
066C FE17C8             cpi ctlw! rz
                  refresh0:
066F E5C5               push h! push b
0671 CD1406             call save$col
0674 2A280D             lhld apos
                  refresh1:
0677 7EB7CA9306         mov a,m! ora a! jz refresh2
067C 4FCDC005DA         mov c,a! call chk$column! jc refresh05
0683 7B32B7FBC3         mov a,e! sta column! jmp refresh2
                  refresh05:
068A E5CD1305           push h! call ctlout
068E E123C37706         pop h! inx h! jmp refresh1
                  refresh2:
0693 3AB7FB322E         lda column! sta new$ctla$col
                  refresh3:
0699 CD1B06             call clear$right
069C CD2B06             call reverse
069F 3A2E0D322D         lda new$ctla$col! sta ctla$column
06A5 C1E1C9             pop b! pop h! ret
                  ;
                  init$apos:
06A8 21250D2228         lxi h,aposi! shld apos
06AE AF322A0D           xra a! sta ctla$sw
06B2 C9                 ret
                  ;
                  init$xpos:
06B3 21C20B2226         lxi h,xbuff! shld xpos! ret
                  ;
                  set$ctla$column:
06BA 212A0D7EB7         lxi h,ctla$sw! mov a,m! ora a! rnz
```

```
06C0 343AB7FB32         inr m! lda column! sta ctla$column! ret
                     ;
                     readi:
06C8 CDC005D482         call chk$column! cnc crlf
06CE 3AC00A47           lda cbuff$len! mov b,a
06D2 0E00CDEE05         mvi c,0! call copy$cbuff
                     else

                     readi:
                        MOV A,D! ORA E! JNZ READ
                        LHLD DMAAD! SHLD INFO
                        INX H! INX H! SHLD CONBUFFADD
                     endif

                     read:  ;read to info address (max length, current length, buffer)

                     if BANKED
06D7 CDB306             call init$xpos
06DA CDA806             call init$apos
                     readx:
06DD CD5E06             call refresh
06E0 AF322B0D           xra a! sta ctlw$sw
                     readx1:

                     endif

06E4 3E0132DFFB         MVI A,1! STA FX
06E9 3AB7FB32BD         lda column! sta strtcol ;save start for ctl-x, ctl-h
06EF 2ADBFB4E23         lhld info! mov c,m! inx h! push h
06F5 AF4732BE0A         XRA A! MOV B,A! STA SAVEPOS
06FA B9C2FF06           CMP C! JNZ $+4
06FE 0C                 INR C
                        ;B = current buffer length,
                        ;C = maximum buffer length,
                        ;HL= next to fill - 1
                        readnx:
                                ;read next character, BC, HL active
06FF C5E5                       push b! push h ;blen, cmax, HL saved
                                readn0:

                     if BANKED
0701 3A2B0DB7CC                         lda ctlw$sw! ora a! cz qconin
                     nxtline:
0708 32BF0A                             sta comchr
                     else
                                        CALL QCONIN ;next char in A
                     endif

                                        ;ani 7fh ;mask parity bit
070B E1C1                               pop h! pop b ;reactivate counters
070D FE0DCAE709                         cpi cr! jz readen ;end of line?
0712 FE0ACAE709                         cpi lf! jz readen ;also end of line

                     if BANKED
0717 FE06C23507                         cpi ctlf! jnz not$ctlf
                                do$ctlf:
```

```
071C CDC0051DBB                 call chk$column! dcr e! cmp e! jnc readnx
                        do$ctlf0:
0724 EB2A280D7E                 xchg! lhld apos! mov a,m! ora a! jz ctlw$115
072D 2322280DEB                 inx h! shld apos! xchg! jmp notr
                        not$ctlf:
0735 FE17C29A07                 cpi ctlw! jnz not$ctlw
                        do$ctlw:
073A EB2A280D7E                 xchg! lhld apos! mov a,m! ora a! jz ctlw$11
0743 EBCDC0051D                 xchg! call chk$column! dcr e! cmp e! xchg! jc ctlw$10
074D EBCD6F06EB                 xchg! call refresh0! xchg! jmp ctlw$113
                        ctlw$10:
0755 2A280D7E                   lhld apos! mov a,m
0759 2322280DC3                 inx h! shld apos! jmp ctlw$13
                        ctlw$11:
0760 212A0D7E36                 lxi h,ctla$sw! mov a,m! mvi m,0
0766 B7CA7307                   ora a! jz ctlw$12
                        ctlw$113:
076A 212B0D3600                 lxi h,ctlw$sw! mvi m,0
                        ctlw$115:
076F EBC3FF06                   xchg! jmp readnx
                        ctlw$12:
0773 3A2B0DB7C2                 lda ctlw$sw! ora a! jnz ctlw$125
077A 78B7C26F07                 mov a,b! ora a! jnz ctlw$115
077F CDB306                     call init$xpos
                        ctlw$125:
0782 2A260D7EB7                 lhld xpos! mov a,m! ora a
0787 322B0DCA6F                 sta ctlw$sw! jz ctlw$115
078D 2322260D                   inx h! shld xpos
                        ctlw$13:
0791 212B0D3617                 lxi h,ctlw$sw! mvi m,ctlw
0796 EBC34D09                   xchg! jmp notr
                        not$ctlw:
079A FE01C2C507                 cpi ctla! jnz not$ctla
                        do$ctla:
                                ;do we have any characters to back over?
079F 3ABD0A573A                 lda strtcol! mov d,a! lda column! cmp d
07A7 CAFF06                     jz readnx
07AA 32BC0A                     sta compcol ;COL > 0
07AD 78B7CA0B09                 mov a,b! ora a! jz linelen
                                ;characters remain in buffer, backup one
07B2 05                         dcr b ;remove one character
                                ;compcol > 0 marks repeat as length compute
                                ;backup one position in xbuff
07B3 E5                         push h
07B4 CDBA06                     call set$ctla$column
07B7 D1                         pop d
07B8 2A280D2B                   lhld apos! dcx h
07BC 22280D1A77                 shld apos! ldax d! mov m,a! xchg! jmp linelen
                        not$ctla:
07C5 FE02C20B08                 cpi ctlb! jnz not$ctlb
                        do$ctlb:
07CA 3ABE0AB8C2                 lda save$pos! cmp b! jnz ctlb$10
07D1 3E17322A0D                 mvi a,ctlw! sta ctla$sw
07D6 32BF0AC33A                 sta comchr! jmp do$ctlw
                        ctlb$10:
07DC EB2A280D04                 xchg! lhld apos! inr b
```

```
                                    ctlb$11:
07E1 053ABE0AB8                             dcr b! lda save$pos! cmp b! jz ctlb$12
07E9 2B1A771BC3                             dcx h! ldax d! mov m,a! dcx d! jmp ctlb$11
                                    ctlb$12:
07F0 22280D                                 shld apos
07F3 C5D5                                   push b! push d
07F5 CDBA06                                 call set$ctla$column
                                    ctlb$13:
07F8 3AB7FB47                               lda column! mov b,a
07FC 3ABD0AB8CA                             lda strtcol! cmp b! jz read$n0
0803 0E08CDCD04                             mvi c,ctin! call conout! jmp ctlb$13
                                    not$ctlb:
080B FE0BC21E08                             cpi ctlK! jnz not$ctlk
0810 EB21250D22                             xchg! lxi h,aposi! shld apos
0817 EBCD5E06                               xchg! call refresh
081B C3FF06                                 jmp readnx
                                    not$ctlk:
081E FE07C22D08                             cpi ctlg! jnz not$ctlg
0823 3A2A0DB7CA                             lda ctla$sw! ora a! jz readnx
082A C32407                                 jmp do$ctlf0
                                    not$ctlg:
                  endif

082D FE08C25008                             cpi ctlh! jnz noth ;backspace?
0832 3ACAFB3CCA                             LDA CTLH$ACT! INR A! JZ DO$RUBOUT
                                    DO$CTLH:
                                            ;do we have any characters to back over?
0839 3ABD0A573A                             LDA STRTCOL! MOV D,A! LDA COLUMN! CMP D
0841 CAFF06                                 jz readnx
0844 32BC0A                                 STA COMPCOL ;COL > 0
0847 78B7CA4D08                             MOV A,B! ORA A! JZ $+4
                                            ;characters remain in buffer, backup one
084C 05                                     dcr b ;remove one character
                                            ;compcol > 0 marks repeat as length compute
084D C30B09                                 jmp linelen ;uses same code as repeat
                                    noth:
                                            ;not a backspace
0850 FE7FC27308                             cpi rubout! jnz notrub ;rubout char?
0855 3ACBFB3CCA                             LDA RUBOUT$ACT! INR A! JZ DO$CTLH
                                    DO$RUBOUT:
                  if BANKED
085C 3E7F32BF0A                             mvi a,rubout! sta comchr
0861 3A2A0DB7C2                             lda ctla$sw! ora a! jnz do$ctlh
                  endif
                                            ;rubout encountered, rubout if possible
0868 78B7CAFF06                             mov a,b! ora a! jz readnx ;skip if len=0
                                            ;buffer has characters, resend last char
086D 7E052B                                 mov a,m! dcr b! dcx h ;A = last char
                                            ;blen=blen-1, next to fill - 1 decremented
0870 C37109                                 jmp rdech1 ;act like this is an echo
                                    notrub:
                                            ;not a rubout character, check end line
0873 FE05C29208                             cpi ctle! jnz note ;physical end line?
                                            ;yes, save active counters and force eol
0878 C57832BE0A                             push b! MOV A,B! STA SAVE$POS
087D E5                                     push h
```

```
                if BANKED
087E 3A2A0DB7C4                       lda ctla$sw! ora a! cnz clear$right
                endif
0885 CD8205                           call crlf
                if BANKED
0888 CD5E06                           call refresh
                endif
088B AF32BD0A                         xra a! sta strtcol ;start position = 00
088F C30107                           jmp readn0 ;for another character
                              notc:
                                      ;not end of line, list toggle?
0892 FE10C2A208                       cpi ctlp! jnz notp ;skip if not ctlp
                                      ;list toggle - change parity
0897 E5                               push h ;save next to fill - 1
0898 C5                               PUSH B
0899 AFCDA004                         XRA A! CALL CONB3
089D C1                               POP B
089E E1C3FF06                         pop h! jmp readnx ;for another char
                              notp:
                                      ;not a ctlp, line delete?
08A2 FE18C2C408                       cpi ctlx! jnz notx
08A7 E1                               pop h ;discard start position
                                      ;loop while column > strtcol
                                      backx:
08A8 3ABD0A21B7                               lda strtcol! lxi h,column
                if BANKED
08AE BEDABD08                                 cmp m! jc backx1
08B2 2A280D7EB7                               lhld apos! mov a,m! ora a! jnz readx
08BA C3D706                                   jmp read
                                          backx1:
                else
                                              cmp m! jnc read ;start again
                endif
08BD 35                                       dcr m ;column = column - 1
08BE CD5705                                   call backup ;one position
08C1 C3A808                                   jmp backx
                              notx:
                                      ;not a control x, control u?
                                      ;not control-X, control-U?
08C4 FE15C2D508                       cpi ctlu! jnz notu ;skip if not
                if BANKED
08C9 E3CDD805E3                       xthl! call copy$xbuff! xthl
                endif
                                      ;delete line (ctlu)
                              do$ctlu:
08CE CD6A05                           call crlfp ;physical eol
08D1 E1                               pop h ;discard starting position
08D2 C3D706                           jmp read ;to start all over
                              notu:
                                      ;not line delete, repeat line?
08D5 FE12C24D09                       cpi ctlr! jnz notr
08DA AF32BE0A                         XRA A! STA SAVEPOS
                if BANKED
08DE EBCDA806EB                       xchg! call init$apos! xchg
08E3 78B7CACE08                       mov a,b! ora a! jz do$ctlu
08E8 EB2A280D04                       xchg! lhld apos! inr b
```

```
                                  ctlr$11:
08ED 05CAF808                             dcr b! jz ctlr$12
08F1 2B1A771B                             dcx h! ldax d! mov m,a! dcx d
08F5 C3ED08                               jmp ctlr$11
                                  ctlr$12:
08F8 22280DC5D5                           shld apos! push b! push d
08FD CD6A053E17                           call crlfp! mvi a,ctlw! sta ctlw$sw
0905 322A0DC301                           sta ctla$sw! jmp readn0
                         endif
                                  linelen:
                                          ;repeat line, or compute line len (ctlh)
                                          ;if compcol > 0
090B C5CD6A05                             push b! call crlfp ;save line length
090F C1E1E5C5                             pop b! pop h! push h! push b
                                          ;bcur, cmax active, beginning buff at HL
                                  rep0:
0913 78B7CA2E09                           mov a,b! ora a! jz rep1 ;count len to 00
0918 234E                                 inx h! mov c,m ;next to print
091A 05                                   DCR B
091B D1D57A9057                           POP D! PUSH D! MOV A,D! SUB B! MOV D,A
0920 C5E5                                 push b! push h ;count length down
0922 3ABE0ABADC                           LDA SAVEPOS! CMP D! CC CTLOUT
0929 E1C1                                 pop h! pop b ;recall remaining count
092B C31309                               jmp rep0 ;for the next character
                                  rep1:
                                          ;end of repeat, recall lengths
                                          ;original BC still remains pushed
092E E5                                   push h ;save next to fill
092F 3ABC0AB7                             lda compcol! ora a ;>0 if computing length
0933 CA0107                               jz readn0 ;for another char if so
                                          ;column position computed for ctlh
0936 21B7FB96                             lxi h,column! sub m ;diff > 0
093A 32BC0A                               sta compcol ;count down below
                                          ;move back compcol-column spaces
                                  backsp:
                                          ;move back one more space
093D CD5705                               call backup ;one space
0940 21BC0A35                             lxi h,compcol! dcr m
0944 C23D09                               jnz backsp
                         if BANKED
0947 CD5E06                               call refresh
                         endif
094A C30107                               jmp readn0 ;for next character
                                  notr:
                                          ;not a ctlr, place into buffer
                                          ;IS BUFFER FULL?
094D F5                                   PUSH A
094E 78B9DA5E09                           MOV A,B! CMP C! JC RDECHO ;NO
                                          ;DISCARD CHARACTER AND RING BELL
0953 F1C5E5                               POP A! PUSH B! PUSH H
0956 0E07CD7AFB                           MVI C,7! CALL CONOUTF! JMP READNO
                                  RDECHO:

                         if BANKED
095E 3ABF0AFE07                           lda comchr! cpi ctlg! jz rdech05
0966 3A2A0DB7C4                           lda ctla$sw! ora a! cnz chk$buffer$size
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
                   rdech05:
        endif

0960 F1                POP A
096E 2377              inx h! mov m,a ;character filled to mem
0970 04                inr b ;blen = blen + 1
                   rdech1:
                       ;look for a random control character
0971 C5E5              push b! push h ;active values saved
0973 4F                mov c,a ;ready to print
        if BANKED
0974 CD1406            call save$col
        endif
0977 CD1305            call ctlout ;may be up-arrow C
097A E1C1              pop h! pop b
        if BANKED
097C 3ABF0AFE07        lda comchr! cpi ctlg! jz do$ctlh
0984 FE7FCA8C09        cpi rubout! jz rdech2
0989 CD5E06            call refresh
                   rdech2:
        endif
098C 3ACFFBE608        LDA CONMODE! ANI 08H! JNZ NOTC
0994 7E                mov a,m ;recall char
0995 FE03              cpi ctlc ;set flags for reboot test
0997 78                mov a,b ;move length to A
0998 C2A009            jnz notc ;skip if not a control c
099B FE01              cpi 1 ;control C, must be length 1
099D CA2903            jz REBOOTX ;reboot if blen = 1
                       ;length not one, so skip reboot
                   notc:
                       ;not reboot, are we at end of buffer?
        if BANKED
09A0 B9D2E009          cmp c! jnc buffer$full
        else
                       jmp readnx ;go for another if not
        endif

        if BANKED
09A4 C5E5              push b! push h
09A6 CDC005DA01        call chk$column! jc readn0
09AC 3A2A0DB7CA        lda ctla$sw! ora a! jz do$new$line
09B3 3ABF0AFE17        lda comchr! cpi ctlw! jz back$one
09BB FE06CAC509        cpi ctlf! jz back$one

                   do$newline:
09C0 3E05C30807        mvi a,ctle! jmp nxtline

                   back$one:
                       ;back up to previous character
09C5 E1C1              pop h! pop b
09C7 05EB              dcr b! xchg
09C9 2A280D2B22        lhld apos! dcx h! shld apos
09D0 1A77EB2B          ldax d! mov m,a! xchg! dcx h
09D4 C5E5CD2B06        push b! push h! call reverse
                       ;disable ctlb or ctlw
09D9 AF322B0DC3        xra a! sta ctlw$sw! jmp readn0
```

```

                                buffer$full:
09E0 AF322B0DC3                         xra a! sta ctlw$sw! jmp readnx
                endif
                                readen:
                                        ;end of read operation, store blen
                if BANKED
09E7 CDC905                             call expand
                endif
09EA E170                               pop h! mov m,b ;M(current len) = B
                if BANKED
09EC C5                                 push b
09ED CDD805                             call copy$xbuff
09F0 C1                                 pop b
09F1 0EFFCDEE05                         mvi c,0ffh! call copy$cbuff
                endif
09F6 21000022BA                         LXI H,0! SHLD CONBUFFADD
09FC 0E0DC3CD04                         mvi c,cr! jmp conout ;return carriage
                                        ;ret
                ;
03C2 =          func1   equ     CONECH
                        ;return console character with echo
                ;
0528 =          func2:  equ     tabout
                        ;write console character with tab expansion
                ;
                func3:
                        ;return reader character
0A01 CD15FF             call readerf
0A04 C3720A             jmp sta$ret
                ;
                ;func4:         equated to punchf
                        ;write punch character
                ;
                ;func5:         equated to listf
                        ;write list character
                        ;write to list device
                ;
                func6:
                        ;direct console i/o - read if 0ffh
0A07 793CCA200A         mov a,c! inr a! jz dirinp ;0ffh => 00h, means input mode
0A0C 3CCA170A                   inr a! JZ DIRSTAT ;0feh => direct STATUS function
0A10 3CCA250A                   INR A! JZ DIRINP1 ;0fdh => direct input, no status
0A14 C37AFB                     JMP CONOUTF
                        DIRSTAT:
                                ;0feH in C for status
0A17 CDFF03C2C0                 CALL CONSTX! JNZ LRET$EQ$FF! JMP STA$RET
                        dirinp:
0A20 CDFF03                     CALL CONSTX ;status check
0A23 B7C8                       ora a! RZ ;sKip, return 00 if not ready
                                ;character is ready, get it
                        dirinp1:
0A25 CDB703                     call CONIN ;to A
0A28 C3720A                     jmp sta$ret
                ;
                func7:
```

```
0A2B CD36FF              call auxinstf
0A2E C3720A              jmp sta$ret
                    ;
                    func8:
0A31 CD39FF              call auxoutstf
0A34 C3720A              jmp sta$ret
                    ;
                    func9:
                         ;write line until $ encountered
0A37 EB                  xchg    ;was lhld info
0A38 4D44                mov c,l! mov b,h ;BC=string address
0A3A C38C05              jmp print ;out to console

06C8 =              func10 equ     readi
                         ;read a buffered console line

                    func11:
                         ;IS CONMODE(1) TRUE?
0A3D 3ACFFB1FD2          LDA CONMODE! RAR! JNC NORMAL$STATUS ;NO
                         ;CTL-C ONLY STATUS CHECK
                    if BANKED
0A44 21D5FB3680          LXI H,QFLAG! MVI M,80H! PUSH H
                    endif
0A4A 21680AE5            LXI H,CTLC$STAT$RET! PUSH H
                         ;DOES KBCHAR = CTL-C?
0A4E 3A20FDFE03          LDA KBCHAR! CPI CTLC! JZ CONB1 ;YES
                         ;IS THERE A READY CHARACTER?
0A56 CD6EFBB7C8          CALL CONSTF! ORA A! RZ ;NO
                         ;IS THE READY CHARACTER A CTL-C?
0A5B CD74FBFE03          CALL CONINF! CPI CTLC! JZ CONB0 ;YES
0A63 3220FDAFC9          STA KBCHAR! XRA A! RET

                    CTLC$STAT$RET:

                    if BANKED
0A68 CD720A              CALL STA$RET
0A6B E13600C9            POP H! MVI M,0! RET
                    else
                         JMP STA$RET
                    endif

                    NORMAL$STATUS:
                         ;check console status
0A6F CD1B04              call conbrk
                         ;(drop through to sta$ret)
                    sta$ret:
                         ;store the A register to aret
0A72 329F0D              sta aret
                    func$ret:                                  ;
0A75 C9                  ret ;jmp goback (pop stack for non cp/m functions)
                    ;
                    setlret1:
                         ;set lret = 1
0A76 3E01C3720A          mvi a,1! jmp sta$ret                  ;
                    ;
                    FUNC109:                     ;GET/SET CONSOLE MODE
```

```
                                ;DOES DE = 0FFFFH?
0A7B 7AA33C                     MOV A,D! ANA E! INR A
0A7E 2ACFFBCA29                 LHLD CONMODE! JZ STHL$RET ;YES - RETURN CONSOLE MODE
0A84 EB22CFFBC9                 XCHG! SHLD CONMODE! RET ;NO - SET CONSOLE MODE
                        ;
                        FUNC110:                        ;GET/SET FUNCTION 9 DELIMITER
0A89 21D3FB                     LXI H,OUT$DELIM
                                ;DOES DE = 0FFFFH?
0A8C 7AA33C                     MOV A,D! ANA E! INR A
0A8F 7ECA720A                   MOV A,M! JZ STA$RET ;YES - RETURN DELIMITER
0A93 73C9                       MOV M,E! RET ;NO - SET DELIMITER
                        ;
                        FUNC111:                        ;PRINT BLOCK TO CONSOLE
                        FUNC112:                        ;LIST BLOCK
0A95 EB5E235623                 XCHG! MOV E,M! INX H! MOV D,M! INX H
0A9A 4E2346EB                   MOV C,M! INX H! MOV B,M! XCHG
                                ;HL = ADDR OF STRING
                                ;BC = LENGTH OF STRING
                        BLK$OUT:
0A9E 78B1C8                     MOV A,B! ORA C! RZ
0AA1 C5E54E                     PUSH B! PUSH H! MOV C,M
0AA4 3ADFFBFE6F                 LDA FX! CPI 111! JZ BLK$OUT1
0AAC CD80FBC3B5                 CALL LISTF! JMP BLK$OUT2
                        BLK$OUT1:
0AB2 CD2B05                     CALL TABOUT
                        BLK$OUT2:
0AB5 E123C10B                   POP H! INX H! POP B! DCX B
0AB9 C39E0A                     JMP BLK$OUT

FB7A =                  SCONOUTF        EQU     CONOUTF


                        ;
                        ;       data areas
                        ;
0ABC 00                 compcol:db      0       ;true if computing column position
0ABD 00                 strtcol:db      0       ;starting column position after read

                        if not BANKED

                        Kbchar:         db      0       ;initial Key char = 00

                        endif

0ABE 00                 SAVEPOS:DB      0       ;POSITION IN BUFFER CORRESPONDING TO
                                                ;BEGINNING OF LINE
                        if BANKED

0ABF 00                 comchr:                 db      0
0AC0 00                 cbuff$len:      db      0
0AC1                    cbuff:          ds      256
0BC1 00                                 db      0
0BC2 00                 xbuff:          db      0
0BC3                                    ds      354
0D25 00                 aposi:          db      0
0D26 0000               xpos:           dw      0
0D28 0000               apos:           dw      0
```

```
0D2A 00         ctla$sw:        db      0
0D2B 00         ctlw$sw:        db      0
0D2C 00         save$column:    db      0
0D2D 00         ctla$column:    db      0
0D2E 00         new$ctla$col:   db      0

                endif

                ;       end of BDOS Console module
                ;
                ;*********************************************************************
                ;****************************************************************
                ;
                ;       Error Messages

                if BANKED

0000 =          md      equ     0

                else

                md      equ     24h

                endif

0D2F 43502F4D20dskmsg:          db      'CP/M Error On '
0D3D 203A2000   dskerr:         db      ' : ',md
0D41 4469736B20permsg:          db      'Disk I/O',md
0D4A 496E76616Cselmsg:          db      'Invalid Drive',md
0D58 526561642Frofmsg:          db      'Read/Only File',md
0D67 526561642Frodmsg:          db      'Read/Only Disk',md

                if not MPM

                passmsg:

                if BANKED
0D76 5061737377         db      'Password Error',md
                endif

                fxstsmsg:
0D85 46696C6520         db      'File Exists',md

                wildmsg:
0D91 3F20696E20         db      '? in Filename',md

                endif
                if MPM

                setlret1:
                        mvi a,1
                sta$ret:
                        sta     aret
                func$ret:
                        ret
                entsp: ds       2
```

```

                        endif

                        ;**************************************************************
                        ;**************************************************************
                        ;
                        ;       common values shared between bdosi and bdos

                        if MPM

                        usrcode:db      0       ; current user number

                        endif

0D9F                    aret:   ds      2       ; address value to return
0D9F =                  lret    equ     aret    ; low(aret)

                        ;**************************************************************
                        ;**************************************************************
                        ;**                                                          **
                        ;**   b a s i c    d i s k    o p e r a t i n g    s y s t e m   **
                        ;**                                                          **
                        ;**************************************************************
                        ;**************************************************************

                        ;       literal constants

00FF =                  true    equ     0ffh    ; constant true
0000 =                  false   equ     000h    ; constant false
FFFF =                  enddir  equ     0ffffh  ; end of directory
0001 =                  byte    equ     1       ; number of bytes for 'byte' type
0002 =                  word    equ     2       ; number of bytes for 'word' type

                        ;       fixed addresses in low memory

005C =                  tfcb    equ     005ch   ; default fcb location
0080 =                  tbuff   equ     0080h   ; default buffer location

                        ;       error message handlers

                        rod$error:
                                ; report read/only disk error
0DA1 0E02C3B20D                 mvi c,2! jmp goerr

                        rof$error:
                                ; report read/only file error
0DA6 0E03C3B20D                 mvi c,3! jmp goerr

                        sel$error:
                                ; report select error
0DAB 0E04                       mvi c,4
                                ; Invalidate curdsk to force select call
                                ; at next curselect call
0DAD 3EFF32A928                 mvi a,0ffh! sta curdsk

                        goerr:
```

```
                        ; hl = .errorhandler, call subroutine
0DB2 612EFF229F         mov h,c! mvi l,0ffh! shld aret

                if MPM
                        call test$error$mode! jnz rtn$phy$errs
                        mov a,c! lxi h,pererr-2! jmp bdos$jmp
                else

                goerr1:
0DB8 3AAA2832ED         lda adrive! sta errdrv
0DBE 3AE7FB3CC4         lda error$mode! inr a! cnz error
                endif

                rtn$phy$errs:

                if MPM
                        lda lock$shell! ora a! jnz lock$perr
                endif

                        ; Return 0ffffh if fx = 27 or 31

0DC5 3ADFFB             lda fx
0DC8 FE1BCA3528         cpi 27! jz goback0
0DCD FE1FCA3528         cpi 31! jz goback0
0DD2 C33B28             jmp goback

                if MPM

                test$error$mode:
                        lxi d,pname+4
                test$error$mode1:
                        call rlr! dad d
                        mov a,m! ani 80h! ret
                endif

                if BANKED

                set$copy$cr$only:
0DD5 3ABC28321D         lda copy$cr$init! sta copy$cr$only! ret

                reset$copy$cr$only:
0DDC AF32BC2832         xra a! sta copy$cr$init! sta copy$cr$only! ret

                endif

                bde$e$bde$m$hl:
0DE4 7B955F             mov a,e! sub l! mov e,a
0DE7 7A9C57             mov a,d! sbb h! mov d,a
0DEA D005C9             rnc! dcr b! ret

                bde$e$bde$p$hl:
0DED 7B855F             mov a,e! add l! mov e,a
0DF0 7A8C57             mov a,d! adc h! mov d,a
0DF3 D004C9             rnc! inr b! ret

                shl3bv:
```

```
0DF6 0C                 inr c
                shl3bv1:
0DF7 0DC8               dcr c! rz
0DF9 298FC3F70D         dad h! adc a! jmp shl3bv1

                incr$rr:
0DFE CD3710             call get$rra
0E01 34C0               inr m! rnz
0E03 2334C0             inx h! inr m! rnz
0E06 2334C9             inx h! inr m! ret

                save$rr:
0E09 CD120EEB           call save$rr2! xchg
                save$rr1:
0E0D 0E03C3290E         mvi c,3! jmp move ; ret
                save$rr2:
0E12 CD371011AE         call get$rra! lxi d,save$ranr! ret

                reset$rr:
0E19 CD120EC30D         call save$rr2! jmp save$rr1 ; ret

                compare:
0E1F 1ABEC0             ldax d! cmp m! rnz
0E22 23130DC8           inx h! inx d! dcr c! rz
0E26 C31F0E             jmp compare

                ;
                ;       local subroutines for bios interface
                ;

                move:
                        ; Move data length of length c from source de to
                        ; destination given by hl
0E29 0C                 inr c ; in case it is zero
                        move0:
0E2A 0DC8                       dcr c! rz ; more to move
0E2C 1A77                       ldax d! mov m,a ; one byte moved
0E2E 1323                       inx d! inx h ; to next byte
0E30 C32A0E                     jmp move0

                selectdisk:
                        ; Select the disk drive given by register D, and fill
                        ; the base addresses curtrka - alloca, then fill
                        ; the values of the disk parameter block
0E33 4A                 mov c,d ; current disk# to c
                        ; lsb of e = 0 if not yet logged - in
0E34 CD1BFF             call seldskf ; hl filled by call
                        ; hl = 0000 if error, otherwise disk headers
0E37 7CB5C8             mov a,h! ora l! rz ; Return with C flag reset if select error
                                ; Disk header block address in hl
0E3A 5E235623                   mov e,m! inx h! mov d,m! inx h ; de=.tran
0E3E 2269282323                 shld cdrmaxa! inx h! inx h ; .cdrmax
0E43 226B282323                 shld curtrka! inx h! inx h ; hl=.currec
0E48 226D282323                 shld curreca! inx h! inx h ; hl=.buffa
0E4D 23226F2823                 inx h! shld drvlbla! inx h
0E52 2271282323                 shld lsn$add! inx h! inx h
```

```
                              ; de still contains .tran
0E57 EB229A28                 xchg! shld tranv ; .tran vector
0E5B 217328                   lxi h,dpbaddr ; de= source for move, hl=dest
0E5E 0E0DCD290E               mvi c,addlist! call move ; addlist filled
                              ; Now fill the disk parameter block
0E63 2A7328EB                 lhld dpbaddr! xchg ; de is source
0E67 218028                   lxi h,sectpt ; hl is destination
0E6A 0E11CD290E               mvi c,dpblist! call move ; data filled
                              ; Now set single/double map mode
0E6F 2A8528                   lhld maxall ; largest allocation number
0E72 7C                       mov a,h ; 00 indicates < 255
0E73 21A32836FF               lxi h,single! mvi m,true ; Assume a=00
0E78 B7CA7E0E                 ora a! jz retselect
                              ; high order of maxall not zero, use double dm
0E7C 3600                     mvi m,false
                      retselect:
                              ; C flag set indicates successful select
0E7E 37C9                     stc! ret

               home:
                      ; Move to home position, then offset to start of dir
0E80 CD18FF           call homef
0E83 AF               xra a ; constant zero to accumulator
0E84 2A6B287723       lhld curtrka! mov m,a! inx h! mov m,a ; curtrk=0000
0E8A 2A6D287723       lhld curreca! mov m,a! inx h! mov m,a ; currec=0000
0E90 2377             inx h! mov m,a ; currec high byte=00

               if MPM
                      lxi h,0! shld dblk ; dblk = 0000
               endif

0E92 C9               ret

               rdbuff:
                      ; Read buffer and check condition
0E93 3E0132B428       mvi a,1! sta readf$sw
0E98 CD27FF           call readf ; current drive, track, sector, dma
0E9B C3A50E           jmp diocomp ; Check for i/o errors

               wrbuff:
                      ; Write buffer and check condition
                      ; write type (wrtype) is in register c
0E9E AF32B428         xra a! sta readf$sw
0EA2 CD2AFF           call writef ; current drive, track, sector, dma
               diocomp: ; Check for disk errors
0EA5 B7C8             ora a! rz
0EA7 4F               mov c,a
0EA8 CDB60E           call chk$media$flag
0EAB 79               mov a,c
0EAC FE03DAB20D       cpi 3! jc goerr
0EB1 0E01C3B20D       mvi c,1! jmp goerr

               chk$media$flag:
                      ; A = 0ffh -> media changed
0EB6 3CC0             inr a! rnz

```

```
                        if BANKED
                                ; Handle media changes as I/O errors for
                                ; permanent drives
0EB8 CDEB0EC8                   call chksiz$eq$8000h! rz
                        endif

                                ; BIOS says media change occurred
                                ; Is disk logged-in?
0EBC 2A6528CD1D                 lhld dlog! call test$vector! mvi c,1! rz ; no - return error
0EC5 CDB211                     call media$change
0EC8 E1                         pop h ; Discard return address
                                ; Was this a flush operation (fx = 48)?
0EC9 3ADFFBFE30                 lda fx! cpi 48! rz ; yes
                                ; Is this a flush to another drive?
0ECF 21AA283AA4                 lxi h,adrive! lda seldsk! cmp m! jnz reset$relog
                                ; Bail out if fx = read, write, close, or search next
0ED9 CD1312                     call chk$exit$fxs
                                ; Is this a directory read operation?
0EDC 3AB428B7C0                 lda readf$sw! ora a! rnz ; yes
                                ; Error - directory write operation
0EE1 0E02C3B20D                 mvi c,2! jmp goerr ; Return disk read/only error

                        reset$relog:
                                ; Reset relog if flushing to another drive
0EE6 AF32DEFBC9                 xra a! sta relog! ret

                        if BANKED

                        chksiz$eq$8000h:
                                ; Return with Z flag set if drive permanent
                                ; with no checksum vector
0EEB 2A8B283E80                 lhld chksiz! mvi a,80h! cmp h! rnz
0EF2 AFBDC9                     xra a! cmp l! ret

                        endif

                        seekdir:
                                ; Seek the record containing the current dir entry

                        if MPM
                                lxi d,0ffffh ; mask = ffff
                                lhld dblk! mov a,h! ora l! jz seekdir1
                                lda blkmsk! mov e,a! xra a! mov d,a ; mask = blkmsk
                                lda blkshf! mov c,a! xra a
                                call shl3bv ; ahl = shl(dblk,blkshf)
                        seekdir1:
                                push h! push a ; Save ahl
                        endif

0EF5 2AE1FB                     lhld dcnt ; directory counter to hl
0EF8 0E02CDAB10                 mvi c,dskshf! call hlrotr ; value to hl
0EFD 229128                     shld drec

                        if MPM

                        ;       arecord = shl(dblk,blkshf) + shr(dcnt,dskshf) & mask
```

```

                            mov a,l! ana e! mov l,a ; dcnt = dcnt & mask
                            mov a,h! ana d! mov h,a
                            pop b! pop d! call bde$e$bde$p$hl

                     else
0F00 0600EB                 mvi b,0! xchg
                     endif

                     set$arecord:
0F03 21AB28                 lxi h,arecord
0F06 7323722370             mov m,e! inx h! mov m,d! inx h! mov m,b
0F0B C9                     ret

                     seek:
                            ; Seek the track given by arecord (actual record)

0F0C 2A6B284E23             lhld curtrka! mov c,m! inx h! mov b,m ; bc = curtrk
0F12 C5                     push b ; s0 = curtrk
0F13 2A6D285E23             lhld curreca! mov e,m! inx h! mov d,m
0F19 2346                   inx h! mov b,m ; bde = currec
0F1B 2AAB283AAD             lhld arecord! lda arecord+2! mov c,a ; chl = arecord
                     seek0:
0F22 7D937C9A79             mov a,l! sub e! mov a,h! sbb d! mov a,c! sbb b
0F28 E5                     push h ; Save low(arecord)
0F29 D2390F                 jnc seek1 ; if arecord >= currec then go to seek1
0F2C 2A8028CDE4             lhld sectpt! call bde$e$bde$m$hl ; currec = currec - sectpt
0F32 E1E32BE3               pop h! xthl! dcx h! xthl ; curtrk = curtrk - 1
0F36 C3220F                 jmp seek0
                     seek1:
0F39 2A8028CDED             lhld sectpt! call bde$e$bde$p$hl ; currec = currec + sectpt
0F3F E1                     pop h ; Restore low(arecord)
0F40 7D937C9A79             mov a,l! sub e! mov a,h! sbb d! mov a,c! sbb b
0F46 DA500F                 jc seek2 ; if arecord < currec then go to seek2
0F49 E323E3                 xthl! inx h! xthl ; curtrk = curtrk + 1
0F4C E5                     push h ; save low (arecord)
0F4D C3390F                 jmp seek1
                     seek2:
0F50 E3E5                   xthl! push h ; hl,s0 = curtrk, s1 = low(arecord)
0F52 2A8028CDE4             lhld sectpt! call bde$e$bde$m$hl ; currec = currec - sectpt
0F58 E1D5C5E5               pop h! push d! push b! push h ; hl,s0 = curtrk,
                            ; s1 = high(arecord,currec), s2 = low(currec),
                            ; s3 = low(arecord)
0F5C EB2A8D2819             xchg! lhld offset! dad d
0F61 444D22FB28             mov b,h! mov c,l! shld track
0F66 CD1EFF                 call settrkf ; call bios settrk routine
                            ; Store curtrk
0F69 D12A6B2873             pop d! lhld curtrka! mov m,e! inx h! mov m,d
                            ; Store currec
0F70 C1D1                   pop b! pop d!
0F72 2A6D287323             lhld curreca! mov m,e! inx h! mov m,d
0F78 2370                   inx h! mov m,b ; currec = bde
0F7A C1                     pop b ; bc = low(arecord), de = low(currec)
0F7B 79936F                 mov a,c! sub e! mov l,a ; hl = bc - de
0F7E 789A67                 mov a,b! sbb d! mov h,a
0F81 CD9D0F                 call shr$physhf
```

```
0F84 444D                    mov b,h! mov c,l

0F86 2A9A28EB                lhld tranv! xchg ; bc=sector#, de=.tran
0F8A CD30FF                  call sectran ; hl = tran(sector)
0F8D 4D44                    mov c,l! mov b,h ; bc = tran(sector)
0F8F 22FD28                  shld sector
0F92 CD21FF                  call setsecf ; sector selected
0F95 2A1BFD4D44              lhld curdma! mov c,l! mov b,h! jmp setdmaf
                             ; ret
                      shr$physhf:
0F9D 3A8F284FC3              lda physhf! mov c,a! jmp hlrotr

                      ;      file control block (fcb) constants

00E5 =                empty  equ     0e5h    ; empty directory entry
007F =                lstrec equ     127     ; last record# on extent
0080 =                recsiz equ     128     ; record size
0020 =                fcblen equ     32      ; file control block size
0004 =                dirrec equ     recsiz/fcblen   ; directory fcbs / record
0002 =                dskshf equ     2       ; log2(dirrec)
0003 =                dskmsk equ     dirrec-1
0005 =                fcbshf equ     5       ; log2(fcblen)

000C =                extnum equ     12      ; extent number field
001F =                maxext equ     31      ; largest extent number
000D =                ubytes equ     13      ; unfilled bytes field
000E =                modnum equ     14      ; data module number

0040 =                maxmod equ     64      ; largest module number

0080 =                fwfmsk equ     80h     ; file write flag is high order modnum
000F =                namlen equ     15      ; name length
000F =                reccnt equ     15      ; record count field
0010 =                dskmap equ     16      ; disk map field
001F =                lstfcb equ     fcblen-1
0020 =                nxtrec equ     fcblen
0021 =                ranrec equ     nxtrec+1; random record field (2 bytes)

                      ;      reserved file indicators

0009 =                rofile equ     9       ; high order of first type char
000A =                invis  equ     10      ; invisible file in dir command

                      ;      utility functions for file access

                      dm$position:
                             ; Compute disk map position for vrecord to hl
0FA4 2182284E                lxi h,blkshf! mov c,m ; shift count to c
0FA8 3AA828                  lda vrecord ; current virtual record to a
                             dmpos0:
0FAB B71F0DC2AB                      ora a! rar! dcr c! jnz dmpos0
                             ; a = shr(vrecord,blkshf) = vrecord/2**(sect/block)
0FB1 47                      mov b,a ; Save it for later addition
0FB2 3E0896                  mvi a,8! sub m ; 8-blkshf to accumulator
0FB5 4F                      mov c,a ; extent shift count in register c
0FB6 3AA628                  lda extval ; extent value ani extmsk
```

```
                     dmpos1:
                             ; blkshf = 3,4,5,6,7, c=5,4,3,2,1
                             ; shift is 4,3,2,1,0
0FB9 0DCAC20F                dcr c! jz dmpos2
0FBD B717C3B90F              ora a! ral! jmp dmpos1
                     dmpos2:
                     ; Arrive here with a = shl(ext and extmsk,7-blkshf)
0FC2 80              add b ; Add the previous shr(vrecord,blkshf) value
                     ; a is one of the following values, depending upon alloc
                     ; bks blkshf
                     ; 1K   3     v/8 + extval * 16
                     ; 2K   4     v/16+ extval * 8
                     ; 4K   5     v/32+ extval * 4
                     ; 8K   6     v/64+ extval * 2
                     ; 16K  7     v/128+extval * 1
0FC3 C9              ret ; with dm$position in a

              getdma:
0FC4 2ADBFB1110      lhld info! lxi d,dskmap! dad d! ret

              getdm:
                     ; Return disk map value from position given by bc
0FCC CDC40F          call getdma
0FCF 09              dad b ; Index by a single byte value
0FD0 3AA328          lda single ; single byte/map entry?
0FD3 B7CADA0F        ora a! jz getdmd ; Get disk map single byte
0FD7 6E60C9                  mov l,m! mov h,b! ret ; with hl=00bb
                     getdmd:
0FDA 09                      dad b ; hl=.fcb(dm+i*2)
                             ; double precision value returned
0FDB 7E23666FC9              mov a,m! inx h! mov h,m! mov l,a! ret

              index:
                     ; Compute disk block number from current fcb
0FE0 CDA40F          call dm$position ; 0...15 in register a
0FE3 329F28          sta dminx
0FE6 4F0600CDCC      mov c,a! mvi b,0! call getdm ; value to hl
0FEC 22AB287DB4      shld arecord! mov a,l! ora h! ret

              atran:
                     ; Compute actual record address, assuming index called

              ;      arecord = shl(arecord,blkshf)

0FF2 3A82284F        lda blkshf! mov c,a
0FF6 2AAB28AFCD      lhld arecord! xra a! call shl3bv
0FFD 22AB2832AD      shld arecord! sta arecord+2

1003 22B128          shld arecord1 ; Save low(arecord)

              ;      arecord = arecord or (vrecord and blkmsk)

1006 3A83284F3A      lda blkmsk! mov c,a! lda vrecord! ana c
100E 47              mov b,a ; Save vrecord & blkmsk in reg b & blk$off
100F 329328          sta blk$off
1012 21AB28B677      lxi h,arecord! ora a! mov m,a! ret
```

```

                get$atts:
                        ; Get volatile attributes starting at f'5
                        ; info locates fcb
1018 2ADBFB             lhld info
101B 11080019           lxi d,8! dad d ; hl = .fcb(f'8)
101F 0E04               mvi c,4
                get$atts$loop:
1021 7E87F5             mov a,m! add a! push a
1024 7A1F57             mov a,d! rar! mov d,a
1027 F10F77             pop a! rrc! mov m,a
102A 2B0DC22110         dcx h! dcr c! jnz get$atts$loop
102F 7AC9               mov a,d! ret

                get$s1:
                        ; Get current s1 field to a
1031 CD3F10237E         call getexta! inx h! mov a,m! ret

                get$rra:
                        ; Get current ran rec field address to hl
1037 2ADBFB1121         lhld info! lxi d,ranrec! dad d ; hl=.fcb(ranrec)
103E C9                 ret

                getexta:
                        ; Get current extent field address to hl
103F 2ADBFB110C         lhld info! lxi d,extnum! dad d ; hl=.fcb(extnum)
1046 C9                 ret

                getrcnta:
                        ; Get reccnt address to hl
1047 2ADBFB110F         lhld info! lxi d,reccnt! dad d! ret

                getfcba:
                        ; Compute reccnt and nxtrec addresses for get/setfcb
104F CD4710EB           call getrcnta! xchg ; de=.fcb(reccnt)
1053 21110019           lxi h,(nxtrec-reccnt)! dad d ; hl=.fcb(nxtrec)
1057 C9                 ret

                getfcb:
                        ; Set variables from currently addressed fcb
1058 CD4F10             call getfcba ; addresses in de, hl
105B 7E32A828           mov a,m! sta vrecord ; vrecord=fcb(nxtrec)
105F EB7EB7C26D         xchg! mov a,m! ora a! jnz getfcb0
1065 CD97144FCD         call get$dir$ext! mov c,a! call set$rc! mov a,m
                getfcb0:
106D FE81DA7410         cpi 81h! jc getfcb1
1072 3E80               mvi a,80h
                getfcb1:
1074 32A528             sta rcount ; rcount=fcb(reccnt) or 80h
1077 CD3F10             call getexta ; hl=.fcb(extnum)
107A 3A8428             lda extmsk ; extent mask to a
107D A6                 ana m ; fcb(extnum) and extmsk
107E 32A628             sta extval
1081 C9                 ret

                setfcb:
```

CP/M - PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
                          ; Place values back into current fcb
1082 CD4F10               call getfcba ; addresses to de, hl
                          ; fcb(cr) = vrecord
1085 3AA82877             lda vrecord! mov m,a
                          ; Is fx < 22? (sequential read or write)
1089 3ADFFBFE16           lda fx! cpi 22! jnc $+4 ; no
                          ; fcb(cr) = fcb(cr) + 1
1091 34                   inr m
1092 EB7EFE80D0           xchg! mov a,m! cpi 80h! rnc ; dont reset fcb(rc) if > 7fh
1097 3AA52877             lda rcount! mov m,a ; fcb(reccnt)=rcount
109B C9                   ret

                  zero$ext$mod:
109C CD3F107223           call getexta! mov m,d! inx h! inx h! mov m,d
10A3 C9                   ret

                  zero:
10A4 70230DC8             mov m,b! inx h! dcr c! rz
10A8 C3A410               jmp zero

                  hlrotr:
                          ; hl rotate right by amount c
10AB 0C                   inr c ; in case zero
10AC 0DC8                 hlrotr0: dcr c! rz ; return when zero
10AE 7CB71F67                     mov a,h! ora a! rar! mov h,a ; high byte
10B2 7D1F6F                       mov a,l! rar! mov l,a ; low byte
10B5 C3AC10                       jmp hlrotr0

                  compute$cs:
                          ; Compute checksum for current directory buffer
10B8 2A6728               lhld buffa ; current directory buffer
10BB 010400               lxi b,4 ; b = 0, c = 4
                  compute$cs0:
10BE 1620                 mvi d,32 ; size of fcb
10C0 AF                   xra a ; clear checksum value
                  compute$cs1:
10C1 862315                       add m! inx h! dcr d
10C4 C2C110                       jnz compute$cs1
10C7 A8470D                       xra b! mov b,a! dcr c
10CA C2BE10                       jnz compute$cs0
10CD C9                   ret ; with checksum in a

                  if MPM

                  compute$cs:
                          ; Compute checksum for current directory buffer
                          mvi c,recsiz ; size of directory buffer
                          lhld buffa ; current directory buffer
                          xra a ; Clear checksum value
                          computecs0:
                                  add m! inx h! dcr c ; cs = cs+buff(recsiz-c)
                                  jnz computecs0
                          ret ; with checksum in a

                  chksum$fcb: ; Compute checksum for fcb
                          ; Add 1st 12 bytes of fcb + curdsk +
          ;     high$ext + xfcb$read$only + bbh
          lxi h,pdcnt! mov a,m
          inx h! add m ; Add high$ext
          inx h! add m ; Add xfcb$read$only
          inx h! add m ; Add curdsk
          adi 0bbh ; Add 0bbh to bias checksum
          lhld info! mvi c,12! call computecs0
          ; Skip extnum
          inx h
          ; Add fcb(s1)
          add m! inx h
          ; Skip modnum
          inx h
          ; Skip fcb(reccnt)
          ; Add disk map
          inx h! mvi c,16! call computecs0
          ora a! ret ; Z flag set if checksum valid

set$chksum$fcb:
          call chksum$fcb! rz
          mov b,a! call gets1
          cma! add b! cma
          mov m,a! ret

reset$chksum$fcb:
          xra a! sta comp$fcb$cks
          call chksum$fcb! rnz
          call get$s1! inr m! ret

endif

check$fcb:

if MPM
          xra a! sta check$fcb4
check$fcb1:
          call chek$fcb! rz
check$fcb2:

          ani 0fh! jnz check$fcb3
          lda pdcnt! ora a! jz check$fcb3
          call set$sdcnt! sta dont$close
          call close1
          lxi h,lret! inr m! jz check$fcb3
          mvi m,0! call pack$sdcnt! mvi b,5
          call search$olist! rz
check$fcb3:

          pop h ; Discard return address
check$fcb4:
          nop
          mvi a,10! jmp sta$ret

set$fcb$cks$flag:
          mvi a,0ffh! sta comp$fcb$cks! ret

```

```
                      else
10CE CD31102A71                 call gets1! lhld lsn$add
10D4 BEC4E310                   cmp m! cnz chK$media$fcb
                      endif

                      cheK$fcb:
10D8 3AB528                     lda high$ext

                      if MPM

                                ; if ext & 0110$0000b = 0110$0000b then
                                ; set fcb(0) to 0 (user 0)

                                cpi 0110$0000b! jnz cheK$fcb1
                      else
10DB B7C8                       ora a! rz
                      endif

10DD 2ADBFBAF77                 lhld info! xra a! mov m,a ; fcb(0) = 0
                      cheK$fcb1:

                      if MPM
                                jmp chKsum$fcb ; ret
                      else
10E2 C9                         ret

                      chK$media$fcb:
                                ; fcb(s1) ^= DPH login sequence # field
                                ; Is fcb addr < bdosadd?

                      if banKed
10E3 2A1EFD                     lhld user$info
                      else
                                lhld info
                      endif

10E6 EB2AFEFBCD                 xchg! lhld bdosadd! call subdh! jnc chK$media1 ; no
                                ; Is rlog(drive) true?
10F0 2A92FBCD1D                 lhld rlog! call testvector! rz ; no
                      chK$media1:
                                ; Return invalid fcb error code
10F7 E1E1                       pop h! pop h
                      chK$media2:
10F9 3E0AC3720A                 mvi a,10! jmp sta$ret
                      endif

                      hlrotl:
                                ; Rotate the mask in hl by amount in c
10FE 0C                         inr c ; may be zero
10FF 0DC8                       hlrotl0: dcr c! rz ; return if zero
1101 29C3FF10                             dad h! jmp hlrotl0

                      set$dlog:
1105 116528                     lxi d,dlog
                      set$cdisK:
                                ; Set a '1' value in curdsK position of bc
```

```
1108 3AA928             lda curdsk
                set$cdisk1:
110B 4F                 mov c,a ; Ready parameter for shift
110C 210100             lxi h,1 ; number to shift
110F CDFE10             call hlrotl ; hl = mask to integrate
1112 1AB51213           ldax d! ora l! stax d! inx d
1116 1AB412C9           ldax d! ora h! stax d! ret

                nowrite:
                        ; Return true if dir checksum difference occurred
111A 2A6328             lhld rodsk

                test$vector:
111D 3AA928             lda curdsk
                test$vector1:
1120 4FCDAB10           mov c,a! call hlrotr
1124 7DE601C9           mov a,l! ani 1b! ret ; non zero if curdsk bit on

                check$rodir:
                        ; Check current directory element for read/only status
1128 CD4011             call getdptra ; address of element

                check$rofile:
                        ; Check current buff(dptr) or fcb(0) for r/o status
112B CD3211             call ro$test
112E D0                 rnc ; Return if not set
112F C3A60D             jmp rof$error ; Exit to read only disk message

                ro$test:
1132 11090019           lxi d,rofile! dad d
1136 7E17C9             mov a,m! ral! ret ; carry set if r/o

                check$write:
                        ; Check for write protected disk
1139 CD1A11C8           call nowrite! rz ; ok to write if not rodsk
113D C3A10D             jmp rod$error ; read only disk error

                getdptra:
                        ; Compute the address of a directory element at
                        ; positon dptr in the buffer

1140 2A67283AB7         lhld buffa! lda dptr
                addh:
                        ; hl = hl + a
1146 856FD0             add l! mov l,a! rnc
                        ; overflow to h
1149 24C9               inr h! ret

                getmodnum:
                        ; Compute the address of the module number
                        ; bring module number to accumulator
                        ; (high order bit is fwf (file write flag)
114B 2ADBFB110E         lhld info! lxi d,modnum! dad d ; hl=.fcb(modnum)
1152 7EC9               mov a,m! ret ; a=fcb(modnum)

                clrmodnum:
```

```
                                ; Clear the module number field for user open/make
 1154 CD4B113600                call getmodnum! mvi m,0 ; fcb(modnum)=0
 1159 C9                        ret

                      clr$ext:
                                ; fcb ext = fcb ext & 1fh
 115A CD3F107EE6                call getexta! mov a,m! ani 0001$1111b! mov m,a!
 1161 C9                        ret

                      setfwf:
 1162 CD4B11                    call getmodnum ; hl=.fcb(modnum), a=fcb(modnum)
                                ; Set fwf (file write flag) to '1'
 1165 F68077                    ori fwfmsk! mov m,a ; fcb(modnum)=fcb(modnum) or 80h
                                ; also returns non zero in accumulator
 1168 C9                        ret

                      compcdr:
                                ; Return cy if cdrmax > dcnt
 1169 2AE1FBEB                  lhld dcnt! xchg ; de = directory counter
 116D 2A6928                    lhld cdrmaxa ; hl=.cdrmax
 1170 7B96                      mov a,e! sub m ; low(dcnt) - low(cdrmax)
 1172 23                        inx h ; hl = .cdrmax+1
 1173 7A9E                      mov a,d! sbb m ; hig(dcnt) - hig(cdrmax)
                                ; condition dcnt - cdrmax  produces cy if cdrmax>dcnt
 1175 C9                        ret

                      setcdr:
                                ; if not (cdrmax > dcnt) then cdrmax = dcnt+1
 1176 CD6911                    call compcdr
 1179 D8                        rc ; Return if cdrmax > dcnt
                                ; otherwise, hl = .cdrmax+1, de = dcnt
 117A 13722B73                  inx d! mov m,d! dcx h! mov m,e
 117E C9                        ret

                      subdh:
                                ; Compute hl = de - hl
 117F 7B956F7A9C                mov a,e! sub l! mov l,a! mov a,d! sbb h! mov h,a
 1185 C9                        ret

                      newchecksum:
 1186 0EFE                      mvi c,0feh ; Drop through to compute new checksum
                      checksum:
                                ; Compute current checksum record and update the
                                ; directory element if c=true, or check for = if not
                                ; drec < chksiz?
 1188 2A9128EB2A                lhld drec! xchg! lhld chksiz
 118F 7CE67F67                  mov a,h! ani 7fh! mov h,a ; Mask off permanent drive bit
 1193 CD7F11                    call subdh ; de-hl
 1196 D0                        rnc ; Skip checksum if past checksum vector size
                                        ; drec < chksiz, so continue
 1197 C5                                push b ; Save init flag
 1198 CDB810                            call compute$cs ; Check sum value to a
 119B 2A7528                            lhld checka ; address of check sum vector
 119E EB                                xchg
 119F 2A9128                            lhld drec
 11A2 19                                dad d ; hl = .check(drec)
```

```
11A3 C1                       pop b ; Recall true=0ffh or false=00 to c
11A4 0C                       inr c ; 0ffh produces zero flag
11A5 CAC611                   jz initial$cs
11A8 0C                       inr c ; 0feh produces zero flag
11A9 CAD111                   jz update$cs

               if MPM
                              inr c! jz test$dir$cs
               endif

                                      ; not initializing, compare
11AC BE                               cmp m ; compute$cs=check(drec)?
11AD C8                               rz ; no message if ok
                                      ; checksum error, are we beyond
                                      ; the end of the disk?
11AE CD1A11C0                         call nowrite! rnz

               media$change:
11B2 CD5212                           call discard$data

               if MPM
                                      call flush$file0
               else
11B5 3EFF32DEFB                       mvi a,0ffh! sta relog! sta hashl
11BD CDE811                           call set$rlog
               endif

                                      ; Reset the drive

11C0 CD0511C35A                       call set$dlog! jmp reset37x

               if MPM
                              test$dir$cs:
                                      cmp m! jnz flush$files
                                      ret
               endif

                              initial$cs:
                                      ; initializing the checksum
11C6 BE77C8                           cmp m! mov m,a! rz
                                      ; or 1 into login seq # if media change
11C9 2A71283E01                       lhld lsn$add! mvi a,1! ora m! mov m,a! ret

                              update$cs:
                                      ; updating the checksum
11D1 77C9                             mov m,a! ret

               set$ro:
                      ; Set current disk to read/only
11D3 3AA4281163       lda seldsk! lxi d,rodsk! call set$cdisk1 ; sets bit to 1
                      ; high water mark in directory goes to max
11DC 2A872823EB       lhld dirmax! inx h! xchg ; de = directory max
11E1 2A6928           lhld cdrmaxa ; hl = .cdrmax
11E4 732372           mov m,e! inx h! mov m,d ; cdrmax = dirmax
11E7 C9               ret
```

```
                      set$rlog:
                                ; rlog(seldsk) = true
11E8 2A90FBCD1D                 lhld olog! call test$vector! rz
11EF 1192FBC308                 lxi d,rlog! jmp set$cdisk

                      tst$log$fxs:
11F5 3A8C28E680                 lda chksiz+1! ani 80h! rnz
11FB 21C028                     lxi h,log$fxs
                      tst$log0:
11FE 3ADFFB47                   lda fx! mov b,a
                      tst$log1:
1202 7EB8C8                     mov a,m! cmp b! rz
1205 23B7C20212                 inx h! ora a! jnz tst$log1
120A 3CC9                       inr a! ret

                      test$media$flag:
120C 2A7128237E                 lhld lsn$add! inx h! mov a,m! ora a! ret

                      chk$exit$fxs:
1213 213B28E5                   lxi h,goback! push h
                                ; does fx = read or write function?
                                ; and is drive removable?
1217 21CD28CDFE                 lxi h,rw$fxs! call tst$log0! jz chk$media2 ; yes
                                ; is fx = close or searchn function?
                                ; and is drive removable?
1220 21D428CDFE                 lxi h,sc$fxs! call tst$log0! jz lret$eq$ff ; yes
1229 E1C9                       pop h! ret

                      tst$relog:
122B 21DEFB7EB7                 lxi h,relog! mov a,m! ora a! rz
1231 3600                       mvi m,0
                      drv$relog:
1233 CD6B1E                     call curselect
1236 21000022E1                 lxi h,0! shld dcnt! xra a! sta dptr
1240 C9                         ret

                      set$lsn:
1241 2A71284E                   lhld lsn$add! mov c,m
1245 CD311071C9                 call getsl! mov m,c! ret

                      discard$data$bcb:
124A 2A7B280E04                 lhld dtabcba! mvi c,4! jmp discard0

                      discard$data:
1252 2A7B28C35B                 lhld dtabcba! jmp discard

                      discard$dir:
1258 2A7928                     lhld dirbcba

                      discard:
125B 0E01                       mvi c,1
                      discard0:
125D 7DA43CC8                   mov a,l! ana h! inr a! rz

                      if BANKED
1261 5E2356EB                   mov e,m! inx h! mov d,m! xchg
```

```
                    discard1:
1265 E5C5                   push h! push b
1267 11AA28CD1F             lxi d,adrive! call compare
126D C1E1C27412             pop b! pop h! jnz discard2

1272 36FF                   mvi m,0ffh
                    discard2:
1274 110D0019               lxi d,13! dad d
1278 5E2356                 mov e,m! inx h! mov d,m
127B EB7DB4C8               xchg! mov a,l! ora h! rz
127F C36512                 jmp discard1
                    else
                            push h
                            lxi d,adrive! call compare
                            pop h! rnz
                            mvi m,0ffh! ret
                    endif

                    get$buffa:
1282 D5110A0019             push d! lxi d,10! dad d
1287 5E2356                 mov e,m! inx h! mov d,m

                    if BANKED
128A 237E32FA28             inx h! mov a,m! sta buffer$bank
                    endif

128F EBD1C9                 xchg! pop d! ret

                    rddir:
                            ; Read a directory entry into the directory buffer
1292 CDF50E                 call seek$dir
1295 3E03C3A212             mvi a,3! jmp wrdir0

                    seek$copy:
                    wrdir:
                            ; Write the current directory entry, set checksum
129A CD3911                 call check$write
129D CD8611                 call newchecksum ; Initialize entry
12A0 3E05                   mvi a,5
                    wrdir0:
12A2 2100002296             lxi h,0! shld last$block
12A8 2A7928                 lhld dirbcba

                    if BANKED
12AB FE05C2B312             cpi 5! jnz $+6
12B0 2AF028                 lhld curbcba
                    endif

12B3 CD5829                 call deblock

                    setdata:
                            ; Set data dma address
12B6 2AD8FBC3BF             lhld dmaad! jmp setdma ; to complete the call

                    setdir1:
12BC CD8212                 call get$buffa
```

```

                          setdma:
                                  ; hl=.dma address to set (i.e., buffa or dmaad)
     12BF 221BFDC9                shld curdma! ret

                          dir$to$user:

                          if not MPM
                                  ; Copy the directory entry to the user buffer
                                  ; after call to search or searchn by user code
     12C3 2A6728EB                lhld buffa! xchg ; source is directory buffer
     12C7 2A19FD                  lhld xdmaad ; destination is user dma address
     12CA 018000                  lxi b,recsiz ; copy entire record
     12CD CD4BFF                  call movef
                          endif
                                  ; Set lret to dcnt & 3 if search successful
     12D0 219F0D7E3C              lxi h,lret! mov a,m! inr a! rz
     12D6 3AE1FBE603              lda dcnt! ani dsKmsk! mov m,a! ret

                          make$fcb$inv: ; Flag fcb as invalid
                                  ; Reset fcb write flag
     12DD CD6211                  call setfwf
                                  ; Set 1st two bytes of disKmap to ffh
     12E0 23233EFF77              inx h! inx h! mvi a,0ffh! mov m,a! inx h! mov m,a
     12E7 C9                      ret

                          chK$inv$fcb: ; Check for invalid fcb
     12E8 CDC40FC3FB              call getdma! jmp test$ffff

                          tst$inv$fcb: ; Test for invalid fcb
     12EE CDE812C0                call chK$inv$fcb! rnz
     12F2 E13E09C372              pop h! mvi a,9! jmp sta$ret! ; lret = 9

                          end$of$dir:
                                  ; Return zero flag if at end of directory, non zero
                                  ; if not at end (end of dir if dcnt = 0ffffh)
     12F8 21E1FB                  lxi h,dcnt
                          test$ffff:
     12FB 7E                      mov a,m ; may be 0ffh
     12FC 23BE                    inx h! cmp m ; low(dcnt) = high(dcnt)?
     12FE C0                      rnz ; non zero returned if different
                                  ; high and low the same, = 0ffh?
     12FF 3C                      inr a ; 0ffh becomes 00 if so
     1300 C9                      ret

                          set$end$dir:
                                  ; Set dcnt to the end of the directory
     1301 21FFFF22E1              lxi h,enddir! shld dcnt! ret

                          read$dir:
     1308 CD0E13C340              call r$dir! jmp r$dir1

                          r$dir:
                                  ; Read next directory entry, with c=true if initializing

     130E 2A8728EB                lhld dirmax! xchg ; in preparation for subtract
```

```
1312 2AE1FB2322         lhld dcnt! inx h! shld dcnt ; dcnt=dcnt+1
                        ; Continue while dirmax >= dcnt (dirmax-dcnt no cy)
1319 CD7F11             call subdh ; de-hl

131C DA0113             jc set$end$dir

                        read$dir0:
                                ; not at end of directory, seek next element
                                ; initialization flag is in c
131F 3AE1FBE603                 lda dcnt! ani dskmsk ; low(dcnt) and dskmsk
1324 0605                       mvi b,fcbshf ; to multiply by fcb size
                                read$dir1:
1326 8705C22613                         add a! dcr b! jnz read$dir1
                                ; a = (low(dcnt) and dskmsk) shl fcbshf
132B 32B728                     sta dptr ; ready for next dir operation
132E B7C0                       ora a! rnz ; Return if not a new record
                        read$dir2:
1330 C5                         push b ; Save initialization flag c
1331 CD9212                     call rd$dir ; Read the directory record
1334 C1                         pop b ; Recall initialization flag
1335 3ADEFBB7C0                 lda relog! ora a! rnz
133A C38811                     jmp checksum ; Checksum the directory elt

                r$dir2:
133D CD3013             call read$dir2
                r$dir1:
1340 3ADEFBB7C8         lda relog! ora a! rz
1345 CD1312             call chk$exit$fxs
1348 CD2B12C392         call tst$relog! jmp rd$dir

                getallocbit:
                        ; Given allocation vector position bc, return with byte
                        ; containing bc shifted so that the least significant
                        ; bit is in the low order accumulator position.  hl is
                        ; the address of the byte for possible replacement in
                        ; memory upon return, and d contains the number of shifts
                        ; required to place the returned value back into position
134E 79E6073C5F         mov a,c! ani 111b! inr a! mov e,a! mov d,a
                        ; d and e both contain the number of bit positions to shift

1354 60690E03           mov h,b! mov l,c! mvi c,3 ; bc = bc shr 3
1358 CDAB10             call hlrotr ; hlrotr does not touch d and e
135B 444D               mov b,h! mov c,l

135D 2A7728             lhld alloca ; base address of allocation vector
1360 097E               dad b! mov a,m ; byte to a, hl = .alloc(bc shr 3)
                        ; Now move the bit to the low order position of a
1362 0710C26213         rotl: rlc! dcr e! jnz rotl! ret

                setallocbit:
                        ; bc is the bit position of alloc to set or reset.  the
                        ; value of the bit is in register e.
1368 D5CD4E13           push d! call getallocbit ; shifted val a, count in d
136C E6FE               ani 1111$1110b ; mask low bit to zero (may be set)
136E C1B1               pop b! ora c ; low bit of c is masked into a
                        ; jmp rotr ; to rotate back into proper position
```

```
                                ; ret

                        rotr:
                                ; byte value from alloc is in register a, with shift count
                                ; in register c (to place bit back into position), and
                                ; target alloc position in registers hl, rotate and replace
1370 0F15C27013                 rrc! dcr d! jnz rotr ; back into position
1375 77                         mov m,a ; back to alloc
1376 C9                         ret

                        copy$alv:
                                ; If Z flag set, copy 1st ALV to 2nd
                                ; Otherwise, copy 2nd ALV to 1st

                        if not BANKED
                                lda bdos$flags! rlc! rlc! rc
                        endif

1377 F5                         push a
1378 CDDE13444D                 call get$nalbs! mov b,h! mov c,l
137D 2A7728545D                 lhld alloca! mov d,h! mov e,l! dad b
1383 F1CA4BFF                   pop a! jz movef
1387 EBC34BFF                   xchg! jmp movef

                        scandm$ab:
                                ; Set/Reset 1st and 2nd ALV
138B C5CDA613                   push b! call scandm$a
138F C1                         pop b! ;jmp scandm$b

                        scandm$b:
                                ; Set/Reset 2nd ALV

                        if not BANKED
                                lda bdos$flags! ani 40h! rnz
                        endif

1390 C5CDDE13                   push b! call get$nalbs
1394 EB2A7728                   xchg! lhld alloca
1398 C1E5192277                 pop b! push h! dad d! shld alloca
139E CDA613                     call scandm$a
13A1 E1227728C9                 pop h! shld alloca! ret

                        scandm$a:
                                ; Set/Reset 1st ALV
                                ; Scan the disk map addressed by dptr for non-zero
                                ; entries, the allocation vector entry corresponding
                                ; to a non-zero entry is set to the value of c (0,1)
13A6 CD4011                     call getdptra ; hl = buffa + dptr
                                ; hl addresses the beginning of the directory entry
13A9 11100019                   lxi d,dskmap! dad d ; hl now addresses the disk map
13AD C5                         push b ; Save the 0/1 bit to set
13AE 0E11                       mvi c,fcblen-dskmap+1 ; size of single byte disk map + 1
                                scandm0:
                                        ; Loop once for each disk map entry
13B0 D1                                 pop d ; Recall bit parity
13B1 0DC8                               dcr c! rz ; all done scanning?
```

```
                                ; no, get next entry for scan
13B3 D5                         push d ; Replace bit parity
13B4 3AA328B7CA                 lda single! ora a! jz scandm1
                                        ; single byte scan operation
13BB C5                                 push b ; Save counter
13BC E5                                 push h ; Save map address
13BD 4E0600                             mov c,m! mvi b,0 ; bc=block#
13C0 C3C913                             jmp scandm2
                                scandm1:
                                        ; double byte scan operation
13C3 0D                                 dcr c ; count for double byte
13C4 C5                                 push b ; Save counter
13C5 4E2346                             mov c,m! inx h! mov b,m ; bc=block#
13C8 E5                                 push h ; Save map address
                                scandm2:
                                        ; Arrive here with bc=block#, e=0/1
13C9 79B0                               mov a,c! ora b ; Skip if = 0000
13CB CAD813                             jz scandm3
13CE 2A8528                             lhld maxall ; Check invalid index
13D1 7D917C98                           mov a,l! sub c! mov a,h! sbb b ; maxall - block#
13D5 D46813                             cnc set$alloc$bit
                                        ; bit set to 0/1
                                scandm3:
13D8 E123                               pop h! inx h ; to next bit position
13DA C1                                 pop b ; Recall counter
13DB C3B013                             jmp scandm0 ; for another item

                get$nalbs: ; Get # of allocation vector bytes
13DE 2A85280E03         lhld maxall! mvi c,3
                        ; number of bytes in allocation vector is (maxall/8)+1
13E3 CDAB1023C9         call hlrotr! inx h! ret

                if MPM

                test$dir:
                        call home
                        call set$end$dir
                test$dir1:
                        mvi c,0feh! call read$dir
                        lda flushed! ora a! rnz
                        call end$of$dir! rz
                        jmp test$dir1
                endif

                initialize:
                        ; Initialize the current disk
                        ; lret = false ; set to true if $ file exists
                        ; Compute the length of the allocation vector - 2

                if MPM
                        lhld tlog! call test$vector! jz initialize1
                        lhld tlog! call remove$drive! shld tlog
                        xra a! sta flushed
                        call test$dir! rz
                initialize1:
                else
```

```
13E8 CD0C1[illegible]600       call test$media$flag! mvi m,0 ; Reset media change flag
13ED CD5212            call discard$data
13F0 CD5812            call discard$dir
                endif

                if BANKED
                       ; Is drive permanent with no chksum vector?
13F3 CDEB0EC205        call chksiz$eq$8000h! jnz initialize2 ; no
                       ; Is this an initial login operation?
                       ; register A = 0
13F9 2A7128BE36        lhld lsn$add! cmp m! mvi m,2! jz initialize2 ; yes
1402 C37713            jmp copy$alv ; Copy 2nd ALV to 1st ALV
                initialize2:

                endif

1405 CDDE14            call get$nalbs ; Get # of allocation vector bytes
1408 444D              mov b,h! mov c,l ; Count down bc til zero
140A 2A7728            lhld alloca ; base of allocation vector
                       ; Fill the allocation vector with zeros
                       initial0:
140D 360023                    mvi m,0! inx h ; alloc(i)=0
1410 0B                        dcx b ; Count length down
1411 78B1C20D14                mov a,b! ora c! jnz initial0

1416 2A6F2877          lhld drvlbla! mov m,a ; Zero out drive desc byte

                       ; Set the reserved space for the directory

141A 2A8928EB          lhld dirblk! xchg
141E 2A7728            lhld alloca ; hl=.alloc()
1421 732372            mov m,e! inx h! mov m,d ; sets reserved directory blks
                       ; allocation vector initialized, home disk
1424 CD800E            call home
                        ; cdrmax = 3 (scans at least one directory record)
1427 2A69283604        lhld cdrmaxa! mvi m,4! inx h! mvi m,0

142F CD0113            call set$end$dir ; dcnt = enddir
1432 2A7D2822B1        lhld hashtbla! shld arecord1

                       ; Read directory entries and check for allocated storage

                       initial2:
1438 0EFFCD0813                mvi c,true! call read$dir
143D CDF812CA77                call end$of$dir! jz copy$alv
                               ; not end of directory, valid entry?
1443 CD4011                    call getdptra ; hl = buffa + dptr
1446 EB2AB1287C                xchg! lhld arecord1! mov a,h! ana l! inr a! xchg
                               ; is hashtbla ^= 0ffffh
144E C48F2B                    cnz init$hash ; yes - call init$hash
1451 3E21BE                    mvi a,21h! cmp m
1454 CA3814                    jz initial2 ; Skip date & time records

1457 3EE5BE                    mvi a,empty! cmp m
145A CA3814                    jz initial2 ; go get another item
```

```
145D 3E20BECA74               mvi a,20h! cmp m! jz drv$lbl
1463 3E10A6C26E               mvi a,10h! ana m! jnz initial3

                              ; Now scan the disk map for allocated blocks

1469 0E01                     mvi c,1 ; set to allocated
146B CDA613                   call scandm$a
                      initial3:
146E CD7611                   call setcdr ; set cdrmax to dcnt
1471 C33B14                   jmp initial2 ; for another entry

                drv$lbl:
1474 110C00197E               lxi d,extnum! dad d! mov a,m
1479 2A6F2877C3               lhld drvlbla! mov m,a! jmp initial3

                copy$dirloc:
                      ; Copy directory location to lret following
                      ; delete, rename, ... ops

1480 3A9D28C372       lda dirloc! jmp sta$ret
                      ; ret

                compext:
                      ; Compare extent# in a with that in c, return nonzero
                      ; if they do not match
1486 C5               push b ; Save c's original value
1487 F53A84282F       push psw! lda extmsk! cma! mov b,a
                      ; b has negated form of extent mask
148D 79A04F           mov a,c! ana b! mov c,a ; low bits removed from c
1490 F1A0             pop psw! ana b ; low bits removed from a
1492 91E61F           sub c! ani maxext ; Set flags
1495 C1               pop b ; Restore original values
1496 C9               ret

                get$dir$ext:
                      ; Compute directory extent from fcb
                      ; Scan fcb disk map backwards
1497 CD4F10           call getfcba ; hl = .fcb(vrecord)
149A 0E10410CC5       mvi c,16! mov b,c! inr c! push b
                      ; b=dskmap pos (rel to 0)
                get$de0:
149F C1               pop b
14A0 0D               dcr c
14A1 AF               xra a ; Compare to zero
                get$de1:
14A2 2B05             dcx h! dcr b; Decr dskmap position
14A4 BEC2AC14         cmp m! jnz get$de2 ; fcb(dskmap(b)) ^= 0
14A8 0DC2A214         dcr c! jnz get$de1
                      ; c = 0 -> all blocks = 0 in fcb disk map
                get$de2:
14AC 79329F28         mov a,c! sta dminx
14B0 3AA328B778       lda single! ora a! mov a,b
14B5 C2B914           jnz get$de3
14B8 1F               rar ; not single, divide blk idx by 2
                get$de3:
14B9 C5E5             push b! push h ; Save dskmap position & count
```

```
14BB 6F2600             mov l,a! mvi h,0 ; hl = non-zero blk idx
                        ; Compute ext offset from last non-zero
                        ; block index by shifting blk idx right
                        ; 7 - blkshf
14BE 3A8228573E         lda blkshf! mov d,a! mvi a,7! sub d
14C5 4FCDAB1045         mov c,a! call hlrotr! mov b,l
                        ; b = ext offset
14CA 3A8428B8E1         lda extmsk! cmp b! pop h! jc get$de0
                        ; Verify computed extent offset <= extmsk
14D2 CD3F104E           call getexta! mov c,m
14D6 2FE61FA1B0         cma! ani maxext! ana c! ora b
                        ; dir ext = (fcb ext & (~ extmsk) & maxext) | ext offset
14DB C1                 pop b ; Restore stack
14DC C9                 ret ; a = directory extent

                searchi:
                        ; search initialization
14DD 2ADBFB22A0         lhld info! shld searcha ; searcha = info
                searchil:
14E3 7932E5FB           mov a,c! sta searchl ; searchl = c
14E7 CDA62B             call set$hash
14EA 3EFF329D28         mvi a,0ffh! sta dirloc ; changed if actually found
14EF C9                 ret

                search$namlen:
14F0 0E0FC3F714         mvi c,namlen! jmp search
                search$extnum:
14F5 0E0C               mvi c,extnum
                search:
                        ; Search for directory element of length c at info
14F7 CDDD14             call searchi
                search1: ; entry point used by rename
14FA CD0113             call set$end$dir ; dcnt = enddir
14FD CDF511CC80         call tst$log$fxs! cz home
                        ; (drop through to searchn)

                searchn:
                        ; Search for the next directory element, assuming
                        ; a previous call on search which sets searcha and
                        ; searchl

                if MPM
                        lxi h,user0$pass! xra a! cmp m! mov m,a! cnz swap
                else
1503 AF32BE28           xra a! sta user0$pass
                endif

1507 CD6B2CC2BA         call search$hash! jnz search$fin
150D 0E00CD0813         mvi c,false! call read$dir ; Read next dir element
1512 CDF812CABA         call end$of$dir! jz search$fin
                                ; not end of directory, scan for match
1518 2AA028EB                   lhld searcha! xchg ; de=beginning of user fcb
151C 1A                         ldax d ; first character
151D FEE5                       cpi empty ; Keep scanning if empty
151F CA2A15                     jz searchnext
                                ; not empty, may be end of logical directory
```

```
1522 D5                          push d ; Save search address
1523 CD6911                      call compcdr ; past logical end?
1526 D1                          pop d ; Recall address
1527 D2BA15                      jnc search$fin ; artificial stop
                 searchnext:
152A CD4011                      call getdptra ; hl = buffa+dptr
152D 3AE5FB4F                    lda searchl! mov c,a ; length of search to c
1531 0600                        mvi b,0 ; b counts up, c counts down

1533 7EFEE5CCED                  mov a,m! cpi empty! cz save$dcnt$pos1

                 if BANKED
1539 AF32A228                    xra a! sta save$xfcb
153D 7EE6EFBECA                  mov a,m! ani 1110$1111b! cmp m! jz search$loop
1544 EBBEEBC257                  xchg! cmp m! xchg! jnz search$loop
154A 3A16FDB7CA                  lda find$xfcb! ora a! jz search$n
1551 32A228C394                  sta save$xfcb! jmp searchoK
                 endif

                                 searchloop:
1557 79B7CA9B15                          mov a,c! ora a! jz endsearch
155C 1AFE3FCA94                          ldax d! cpi '?'! jz searchoK ; ? in user fcb
                                         ; Scan next character if not ubytes
1562 78FE0DCA94                          mov a,b! cpi ubytes! jz searchoK
                                         ; not the ubytes field, extent field?
1568 FE0C                                cpi extnum ; may be extent field
156A CA7C15                              jz searchext ; SKip to search extent
156D FE0E1ACCB7                          cpi modnum! ldax d! cz searchmod
1573 96E67F                              sub m! ani 7fh ; Mask-out flags/extent modulus
1576 C2C715                              jnz searchnm ; SKip if not matched
1579 C39415                              jmp searchoK ; matched character
                                 searchext:
157C 1A                                  ldax d
                                         ; Attempt an extent # match
157D C5                                  push b ; Save counters

                 if MPM
                                         push h
                                         lhld sdcnt
                                         inr h! jnz dont$save
                                         lhld dcnt! shld sdcnt
                                         lhld dblK! shld sdblK
                                 dont$save:
                                         pop h
                 endif

157E 4E                                  mov c,m ; directory character to c
157F CD8614                              call compext ; Compare user/dir char

1582 47                                  mov b,a
1583 3ABE283CCA                          lda user0pass! inr a! jz save$dcnt$pos2
                                         ; Disable search of user 0 if any fcb
                                         ; is found under the current user #
158A AF32BD28                            xra a! sta search$user0
158E 78                                  mov a,b

```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
158F C1                                   pop b ; Recall counters
1590 B7                                   ora a ; Set flag
1591 C20315                               jnz searchn ; Skip if no match
                                  searchok:
                                          ; current character matches
1594 1323040D                             inx d! inx h! inr b! dcr c
1598 C35715                               jmp searchloop
                                  endsearch:
                                          ; entire name matches, return dir position

                  if BANKED
159B 3AA2283CC2                           lda save$xfcb! inr a! jnz endsearch1
15A2 3A18FDFEFE                           lda xdcnt+1! cpi 0feh! cz save$dcnt$pos0
15AA C30315                               jmp searchn
                                  endsearch1:
                  endif

15AD AF329D28                             xra a! sta dirloc ; dirloc = 0
15B1 329F0D                               sta lret ; lret = 0
                                          ; successful search -
                                          ; return with zero flag reset
15B4 4704C9                               mov b,a! inr b! ret
                                  searchmod:
15B7 E63FC9                               ani 3fh! ret ; Mask off high 2 bits
                                  search$fin:
                                          ; end of directory, or empty name

15BA CDED15                               call save$dcnt$pos1

                                          ; Set dcnt = 0ffffh
15BD CD0113                               call set$end$dir ; may be artifical end
                                  lret$eq$ff:
                                          ; unsuccessful search -
                                          ; return with zero flag set
                                          ; lret,low(aret) = 0ffh
15C0 3EFF4704C3                           mvi a,255! mov b,a! inr b! jmp sta$ret

                                  searchnm: ; search no match routine
15C7 78B7C20315                           mov a,b! ora a! jnz searchn ; fcb(0)?
15CC 7EB7C20315                           mov a,m! ora a! jnz searchn ; dir fcb(0)=0?
15D1 3ABD28B7CA                           lda search$user0! ora a! jz searchn
15D8 32BE28                               sta user0$pass

                  if MPM
                                          call swap
                  endif

15DB C39415                               jmp searchok

                  if MPM

                  swap: ; Swap dcnt,sdblk with sdcnt0,sdblk0
                          push h! push d! push b
                          lxi d,sdcnt! lxi h,sdcnt0
                          mvi b,4
                  swap1:
                      ldax d! mov c,a! mov a,m
                      stax d! mov m,c
                      inx h! inx d! dcr b! jnz swap1
                      pop b! pop d! pop h!
                      ret
               endif

               save$dcnt$pos2:
                      ; Save directory position of matching fcb
                      ; under user 0 with matching extent # & modnum = 0
                      ; a = 0 on entry
15DE B0C1010315       ora b! pop b! lxi b,searchn! push b! rnz
15E5 23237EB7C0       inx h! inx h! mov a,m! ora a! rnz
                      ; Call if user0pass = 0ffh &
                      ;         dir fcb(extnum) = fcb(extnum)
                      ;         dir fcb(modnum) = 0
               save$dcnt$pos0:
15EA CDF515           call save$dcnt$pos ; Return to searchn
               save$dcnt$pos1:
                      ; Save directory position of first empty fcb
                      ; or the end of the directory
15ED E5               push h!
15EE 2A17FD24C2       lhld xdcnt! inr h! jnz save$dcnt$pos$ret ; Return if h ^= 0ffh
               save$dcnt$pos:
15F5 2AE1FB2217       lhld dcnt! shld xdcnt

               if MPM
                      lhld dblk! shld xdblk
               endif

               save$dcnt$pos$ret:
15FB E1C9             pop h! ret

               if BANKED

               init$xfcb$search:
15FD 3EFF             mvi a,0ffh
               init$xfcb$search1:
15FF 3216FD3EFE       sta find$xfcb! mvi a,0feh! sta xdcnt+1! ret

               does$xfcb$exist:
1608 3A18FDFEFE       lda xdcnt+1! cpi 0feh! rz
160E CD0A19           call set$dcnt$dblk
1611 AFCDFF15         xra a! call init$xfcb$search1
1615 2AA0287EF6       lhld searcha! mov a,m! ori 10h! mov m,a
161C 0E0CCDE314       mvi c,extnum! call searchi1! jmp searchn

               xdcnt$eq$dcnt:
1624 2AE1FB2217       lhld dcnt! shld xdcnt! ret

               restore$dir$fcb:
162B CD0A19           call set$dcnt$dblk
162E 0E0FCDDD14       mvi c,namlen! call searchi! jmp searchn
               endif

               delete:
```

```
                                ; Delete the currently addressed file
1636 CD1810                     call get$atts

                      if BANKED
1639 32B328                     sta attributes
                                ; Make search return matching fcbs and xfcbs
                      deletex:
163C 3EFECDFF15                 mvi a,0feh! call init$xfcb$search1
                      else
                                ; Return with aret = 0 for XFCB only delete
                                ; in non-banked systems
                                ral! rc
                      endif

                      ; Delete pass 1 - check r/o attributes and xfcb passwords

1641 CDF514C8                   call search$extnum! rz

                         delete00:
1645 CA8416                           jz delete1

                      if BANKED
                                      ; Is addressed dir fcb an xfcb?
1648 CD40117EE6                       call getdptra! mov a,m! ani 10h! jnz delete01 ; yes

                      if MPM
                                      call tst$olist ; Verify fcb not open by someone else
                      endif

                                      ; Check r/o attribute if this is not an
                                      ; xfcb only delete operation.
1651 3AB32817D4                       lda attributes! ral! cnc check$rodir
                      else
                                      call check$rodir
                      endif

                      if BANKED
                                      ; Are xfcb passwords enabled?
1658 CD731F17DA                       call get$dir$mode! ral! jc delete02 ; no
                      endif

                                      ; Is this a wild card delete operation?
165F 2ADBFBCD39                       lhld info! call chk$wild! jz delete02 ; yes
                                      ; Not wild & passwords inactive
                                      ; Skip to pass 2
1668 C38A16                           jmp delete11

                      if BANKED

                         delete01:
                                      ; Check xfcb password if passwords enabled
166B CD731F17D2                       call get$dir$mode! ral! jnc delete02
1672 CDD420CA7E                       call chk$xfcb$password! jz delete02
1678 CD851FC33C                       call chk$pw$error! jmp deletex
                      endif

```

```
                   delete02:
167E CD0315C345            call searchn! jmp delete00

               ; Delete pass 2 - delete all matching fcbs and/or xfcbs.

               delete1:
1684 CDF514           call search$extnum

                   delete10:
1687 CA8014                jz copy$dir$loc
                   delete11:
168A CD4011                call getdptra

               if BANKED
                           ; Is addressed dir fcb an xfcb?
168D 7EE610C29B            mov a,m! ani 10h! jnz delete12 ; yes
               if MPM
                           push h
                           call chK$olist ; Delete olist item if present
                           pop h
               endif
                           ; Is this delete operation xfcb only?
1693 3AB328E680            lda attributes! ani 80h! jnz delete13 ; yes
               endif

                   delete12:
                           ; Delete dir fcb or xfcb
                           ; if fcb free all alocated blocks.

169B 36E5                  mvi m,empty

               if BANKED

                   delete13:
169D F5                    push a ; Z flag set => free FCB blocks
                           ; Zero password mode byte in sfcb if sfcb exists
                           ; Does sfcb exist?
169E CD1021B7C2            call get$dtba$8! ora a! jnz $+4 ; no
                           ; Zero mode byte
16A5 77                    mov m,a
               endif

16A6 CD9A120E00            call wrdir! mvi c,0

               if BANKED
16AB F1CC8B13              pop a! cz scandm$ab
               else
                           call scandm$ab
               endif

16AF CDC82C                call fix$hash
16B2 CD0315C387            call searchn! jmp delete10

               get$block:
                      ; Given allocation vector position bc, find the zero bit
                      ; closest to this position by searching left and right.
```

```
                     ; if found, set the bit to one and return the bit position
                     ; in hl.  if not found (i.e., we pass 0 on the left, or
                     ; maxall on the right), return 0000 in hl
16B8 5059            mov d,b! mov e,c ; copy of starting position to de
                     righttst:
16BA 2A8528                  lhld maxall ; value of maximum allocation#
16BD 7B957A9C                mov a,e! sub l! mov a,d! sbb h ; right=maxall?
16C1 D2EE16                  jnc retblock0 ; return block 0000 if so
16C4 13C5D5                  inx d! push b! push d ; left, right pushed
16C7 424B                    mov b,d! mov c,e ; ready right for call
16C9 CD4E13                  call getallocbit
16CC 1FD2E616                rar! jnc retblock ; Return block number if zero
16D0 D1C1                    pop d! pop b ; Restore left and right pointers
                     lefttst:
16D2 79B0CABA16              mov a,c! ora b! jz righttst ; Skip if left=0000
                             ; left not at position zero, bit zero?
16D7 0BD5C5                  dcx b! push d! push b ; left,right pushed
16DA CD4E13                  call getallocbit
16DD 1FD2E616                rar! jnc retblock ; return block number if zero
                             ; bit is one, so try the right
16E1 C1D1                    pop b! pop d ; left, right restored
16E3 C3BA16                  jmp righttst
                     retblock:
16E6 173C                    ral! inr a ; bit back into position and set to 1
                             ; d contains the number of shifts required to reposition
16E8 CD7013                  call rotr ; move bit back to position and store
16EB E1D1                    pop h! pop d ; hl returned value, de discarded
16ED C9                      ret
                     retblock0:
                             ; cannot find an available bit, return 0000
16EE 79                      mov a,c
16EF B0C2D216                ora b! jnz lefttst ; also at beginning
16F3 210000C9                lxi h,0000h! ret

              copy$dir:
                     ; Copy fcb information starting at c for e bytes
                     ; into the currently addressed directory entry
16F7 1680            mvi d,80h
              copy$dir0:
16F9 CD1017          call copy$dir2
16FC 0C              inr c
              copy$dir1:
16FD 0DCA9A12        dcr c! jz seek$copy
1701 7EA0C5          mov a,m! ana b! push b
1704 471AE67FB0      mov b,a! ldax d! ani 7fh! ora b! mov m,a
170A C12313C3FD      pop b! inx h! inx d! jmp copy$dir1
              copy$dir2:
1710 D5              push d ; Save length for later
1711 0600            mvi b,0 ; double index to bc
1713 2ADBFB          lhld info ; hl = source for data
1716 09              dad b
1717 237ED624CC      inx h! mov a,m! sui '$'! cz set$submit$flag
171E 2BEB            dcx h! xchg ; de=.fcb(c), source for copy
1720 CD4011          call getdptra ; hl=.buff(dptr), destination
1723 C1              pop b ; de=source, hl=dest, c=length
1724 C9              ret
```

```

                     set$submit$flag:
1725 11B3FB1AF6              lxi d,ccp$flgs! ldax d! ori 1! stax d! ret

                     check$wild:
                             ; Check for ? in file name or type
172D 2ADBFB                  lhld info
                     check$wild0:   ; entry point used by rename
1730 CD3917C0                call chk$wild! rnz
1734 3E09C3E523              mvi a,9! jmp set$aret

                     chk$wild:
1739 0E0B                    mvi c,11
                     chk$wild1:
173B 233E3F96E6              inx h! mvi a,3fh! sub m! ani 7fh! rz
1742 0DC23B17B7              dcr c! jnz chk$wild1! ora a! ret

                     copy$user$no:
1748 2ADBFB7E01              lhld info! mov a,m! lxi b,dskmap
174F 0977C9                  dad b! mov m,a! ret

                     rename:
                             ; Rename the file described by the first half of
                             ; the currently addressed file control block. The
                             ; new name is contained in the last half of the
                             ; currently addressed file control block.  The file
                             ; name and type are changed, but the reel number
                             ; is ignored.  The user number is identical.

                             ; Verify that the new file name does not exist.
                             ; Also verify that no wild chars exist in
                             ; either filename.

                     if MPM
                             call getatts! sta attributes
                     endif

                             ; Verify that no wild chars exist in 1st filename.
1752 CD2D17                  call check$wild

                     if BANKED
                             ; Check password of file to be renamed.
1755 CD781FC485              call chk$password! cnz chk$pw$error
                             ; Setup search to scan for xfcbs.
175B CDFD15                  call init$xfcb$search
                     endif

                             ; Copy user number to 2nd filename
175E CD4817                  call copy$user$no
1761 22A028                  shld searcha

                             ; Verify no wild chars exist in 2nd filename
1764 CD3017                  call check$wild0

                             ; Verify new filename does not already exist
1767 0E0C2AA028              mvi c,extnum! lhld searcha! call searchi! call search1
```

```
1772 C2E323             jnz file$exists ; New filename exists

                if BANKED
                        ; If an xfcb exists for the new filename, delete it.
1775 CD0816C48A         call does$xfcb$exist! cnz delete11
                endif

177B CD4817             call copy$user$no

                if BANKED
177E CDFD15             call init$xfcb$search
                endif

                        ; Search up to the extent field
1781 CDF514             call search$extnum
1784 C8                 rz
1785 CD2811             call check$rodir ; may be r/o file

                if MPM
                        call chK$olist
                endif

                        ; Copy position 0
                        rename0:
                                ; not end of directory, rename next element
1788 0E101E0CCD                 mvi c,dsKmap! mvi e,extnum! call copy$dir
                                ; element renamed, move to next

178F CDC82C                     call fix$hash
1792 CD0315                     call searchn
1795 C28817                     jnz rename0
                        rename1:

                if BANKED
1798 CD0816CA80                 call does$xfcb$exist! jz copy$dir$loc
179E CD4817C388                 call copy$user$no! jmp rename0
                else
                                jmp copy$dir$loc
                endif

                indicators:
                        ; Set file indicators for current fcb
17A4 CD1810             call get$atts ; Clear f5' through f8'
17A7 32B328             sta attributes

                if BANKED
17AA CD781FC485         call chK$password! cnz chK$pw$error
                endif

17B0 CDF514             call search$extnum ; through file type
17B3 C8                 rz

                if MPM
                        call chK$olist
                endif
```

```
                              indic0:
                                      ; not end of directory, continue to change
17B4 0E001E0C                         mvi c,0! mvi e,extnum ; Copy name
17B8 CD1017CD29                       call copy$dir2! call move
17BE 3AB328E640                       lda attributes! ani 40h! jz indic1

                                      ; If interface att f6' set, dir fcb(s1) = fcb(cr)

17C6 E5CD4F107E                       push h! call getfcba! mov a,m
17CB E12377                           pop h! inx h! mov m,a
                              indic1:
17CE CD9A12                           call seek$copy
17D1 CD0315                           call searchn
17D4 CA8014                           jz copy$dir$loc
17D7 C3B417                           jmp indic0

                      open:
                              ; Search for the directory entry, copy to fcb
17DA CDF014                   call search$namlen
                      open1:
17DD C8                       rz ; Return with lret=255 if end
                              ; not end of directory, copy fcb information
                      open$copy:
17DE CD62115FE5               call setfwf! mov e,a! push h! dcx h! dcx h
17E5 56D5                     mov d,m! push d ; Save extent# & module# with fcb write flag set
17E7 CD4011EB                 call getdptra! xchg ; hl = .buff(dptr)
17EB 2ADBFB                   lhld info ; hl=.fcb(0)
17EE 0E20                     mvi c,nxtrec ; length of move operation
17F0 CD290E                   call move ; from .buff(dptr) to .fcb(0)
                              ; Note that entire fcb is copied, including indicators
17F3 CD97144F                 call get$dir$ext! mov c,a
                              ; Restore module # and extent #
17F7 D1E1732B2B               pop d! pop h! mov m,e! dcx h! dcx h! mov m,d
                              ; hl = .user extent#, c = dir extent#
                              ; above move set fcb(reccnt) to dir(reccnt)
                              ; if fcb ext < dir ext then fcb(reccnt) = fcb(reccnt) | 128
                              ; if fcb ext = dir ext then fcb(reccnt) = fcb(reccnt)
                              ; if fcb ext > dir ext then fcb(reccnt) = 0

                      set$rc: ; hl=.fcb(ext), c=dirext
17FD 0600                             mvi b,0
17FF EB21030019                       xchg! lxi h,(reccnt-extnum)! dad d
                                      ; Is fcb ext = dirext?
1804 1A91CA1218                       ldax d! sub c! jz set$rc2 ; yes
                                      ; Is fcb ext > dirext?
1809 78D21018                         mov a,b! jnc set$rc1 ; yes - fcb(rc) = 0
                                      ; fcb ext  < dirext
                                      ; fcb(rc) = 128 | fcb(rc)
180D 3E80B6                           mvi a,128! ora m
                              set$rc1:
1810 77C9                             mov m,a! ret
                              set$rc2:
                                      ; fcb ext = dirext
1812 7EB7C0                           mov a,m! ora a! rnz ; ret if fcb(rc) ^= 0
                              set$rc3:
1815 3600                             mvi m,0 ; required by function 99
```

```
1817 3A9F28B7C8                 lda dminx! ora a! rz ; ret if no blks in fcb
181C 3680C9                     mvi m,128! ret ; fcb(rc) = 128

                mergezero:
                        ; hl = .fcb1(i), de = .fcb2(i),
                        ; if fcb1(i) = 0 then fcb1(i) := fcb2(i)
181F 7E23B62BC0         mov a,m! inx h! ora m! dcx h! rnz ; return if = 0000
1824 1A771323           ldax d! mov m,a! inx d! inx h ; low byte copied
1828 1A771B2B           ldax d! mov m,a! dcx d! dcx h ; back to input form
182C C9                 ret

                restore$rc:
                        ; hl = .fcb(extnum)
                        ; if fcb(rc) > 80h then fcb(rc) = fcb(rc) & 7fh
182D E5                 push h
182E 11030019           lxi d,(reccnt-extnum)! dad d
1832 7EFE81DA3B         mov a,m! cpi 81h! jc restore$rc1
1838 E67F77             ani 7fh! mov m,a
                restore$rc1:
183B E1C9               pop h! ret

                close:
                        ; Locate the directory element and re-write it
183D AF329F0D           xra a! sta lret

                if MPM
                        sta dont$close
                endif

1841 CD1A11C0           call nowrite! rnz ; Skip close if r/o disk
                        ; Check file write flag - 0 indicates written
1845 CD4B11             call getmodnum ; fcb(modnum) in a
1848 E680C0             ani fwfmsk! rnz ; Return if bit remains set
                close1:
184B CDE812CAF2         call chk$inv$fcb! jz mergerr

                if MPM
                        call set$fcb$cks$flag
                endif

1851 CD97144F           call get$dir$ext! mov c,a
1855 46C5               mov b,m! push b
                        ; b = original extent, c = directory extent
                        ; Set fcb(ex) to directory extent
1857 71                 mov m,c
                        ; Recompute fcb(rc)
1858 CD2D18             call restore$rc
                        ; Call set$rc if fcb ext > dir ext
185B 79B8DCFD17         mov a,c! cmp b! cc set$rc
1860 CD6C18             call close$fcb
                        ; Restore original extent & reset fcb(rc)
1863 CD3F10C1           call get$exta! pop b
1867 4E70C3FD17         mov c,m! mov m,b! jmp set$rc ; Reset fcb(rc)

                close$fcb:
                        ; Locate file
```

```
186C CDF014           call search$namlen
186F C8               rz ; Return if not found
                      ; Merge the disk map at info with that at buff(dptr)
1870 011000CD67       lxi b,dskmap! call get$fcb$adds
1876 0E10             mvi c,(fcblen-dskmap) ; length of single byte dm
                      merge0:
1878 3AA328B7CA               lda single! ora a! jz merged ; Skip to double
                              ; This is a single byte map
                              ; if fcb(i) = 0 then fcb(i) = buff(i)
                              ; if buff(i) = 0 then buff(i) = fcb(i)
                              ; if fcb(i) <> buff(i) then error
187F 7EB71AC286               mov a,m! ora a! ldax d! jnz fcbnzero
                                      ; fcb(i) = 0
1885 77                               mov m,a ; fcb(i) = buff(i)
                              fcbnzero:
1886 B7C28C18                 ora a! jnz buffnzero
                                      ; buff(i) = 0
188A 7E12                             mov a,m! stax d ; buff(i)=fcb(i)
                              buffnzero:
188C BEC2F218                 cmp m! jnz mergerr ; fcb(i) = buff(i)?
1890 C3A818                   jmp dmset ; if merge ok
                      merged:
                              ; This is a double byte merge operation
1893 CD1F18                   call mergezero ; buff = fcb if buff 0000
1896 EBCD1F18EB               xchg! call mergezero! xchg ; fcb = buff if fcb 0000
                              ; They should be identical at this point
189B 1ABEC2F218               ldax d! cmp m! jnz mergerr ; low same?
18A0 1323                     inx d! inx h ; to high byte
18A2 1ABEC2F218               ldax d! cmp m! jnz mergerr ; high same?
                              ; merge operation ok for this pair
18A7 0D                       dcr c ; extra count for double byte
                      dmset:
18A8 1323                     inx d! inx h ; to next byte position
18AA 0DC27818                 dcr c! jnz merge0 ; for more
                              ; end of disk map merge, check record count
                              ; de = .buff(dptr)+32, hl = .fcb(32)

18AE EB01ECFF09               xchg! lxi b,-(fcblen-extnum)! dad b! push h
18B4 CD9714D1                 call get$dir$ext! pop d

                              ; hl = .fcb(extnum), de = .buff(dptr+extnum)

18B8 CDF818                   call compare$extents

                              ; b=1 -> fcb(ext) ^= dir ext = buff(ext)
                              ; b=2 -> fcb(ext) = dir ext ^= buff(ext)
                              ; b=3 -> fcb(ext) = dir ext = buff(ext)

                              ; fcb(ext), buff(ext) = dir ext
18BB 7712C5                   mov m,a! stax d! push b

18BE 01030009EB               lxi b,(reccnt-extnum)! dad b! xchg! dad b
18C4 C1                       pop b

                              ; hl = .buff(rc) , de = .fcb(rc)
```

```
18C5 05CAD918                 dcr b! jz mrg$rc1 ; fcb(rc) = buff(rc)

18C9 05CADA18                 dcr b! jz mrg$rc2 ; buff(rc) = fcb(rc)

18CD 1ABEDAD918               ldax d! cmp m! jc mrg$rc1 ; Take larger rc
18D2 B7C2DA18                 ora a! jnz mrg$rc2
18D6 CD1518                   call set$rc3

18D9 EB               mrg$rc1: xchg

18DA 1A77             mrg$rc2: ldax d! mov m,a

               if MPM
                              lda dont$close! ora a! rnz
               endif

                              ; Set t3' off indicating file update
18DC CD4011110B               call getdptra! lxi d,11! dad d
18E3 7EE67F77                 mov a,m! ani 7fh! mov m,a
18E7 CD6211                   call setfwf
18EA 0E01CD9013               mvi c,1! call scandm$b ; Set 2nd ALV vector
18EF C39A12                   jmp seek$copy ; OK to 'wrdir' here - 1.4 compat
                              ; ret
                      mergerr:
                              ; elements did not merge correctly
18F2 CDDD12                   call make$fcb$inv
18F5 C3C015                   jmp lret$eq$ff

               compare$extents:
18F8 0601BEC0         mvi b,1! cmp m! rnz
18FC 04EBBEEBC0       inr b! xchg! cmp m! xchg! rnz
1901 04C9             inr b! ret

               set$xdcnt:
1903 21FFFF2217       lxi h,0ffffh! shld xdcnt! ret

               set$dcnt$dblk:
190A 2A17FD           lhld xdcnt
               set$dcnt$dblk1:
190D 3EFCA5           mvi a,1111$1100b! ana l
1910 6F2B22E1FB       mov l,a! dcx h! shld dcnt

               if MPM
                      lhld xdblk! shld dblk
               endif

1915 C9               ret

               if MPM

               sdcnt$eq$xdcnt:
                      lxi h,sdcnt! lxi d,xdcnt! mvi c,4
                      jmp move
               endif

               make:
```

```
                        ; Create a new file by creating a directory entry
                        ; then opening the file

1916 2117FDCDFB         lxi h,xdcnt! call test$ffff! cnz set$dcnt$dblk

191F 2ADBFBE5           lhld info! push h ; Save fcb address, Look for E5
1923 21622822DB         lxi h,efcb! shld info ; info = .empty
1929 0E01               mvi c,1

192B CDDD14CD03         call searchi! call searchn

                        ; zero flag set if no space
1931 E1                 pop h ; Recall info address
1932 22DBFB             shld info ; in case we return here
1935 C8                 rz ; Return with error condition 255 if not found

                if BANKED
                        ; Return early if making an xfcb
1936 3A15FDB7C0         lda make$xfcb! ora a! rnz
                endif

                        ; Clear the remainder of the fcb
                        ; Clear s1 byte
193B 110D001972         lxi d,13! dad d! mov m,d! inx h
                        ; Clear and save file write flag of modnum
1941 7EF5E5E63F         mov a,m! push a! push h! ani 3fh! mov m,a! inx h
1948 3E01               mvi a,1
194A 0E11               mvi c,fcblen-namlen ; number of bytes to fill
                        make0:
194C 72230DC24C                 mov m,d! inx h! dcr c! jnz make0
1952 3D4ACC1221                 dcr a! mov c,d! cz get$dtba
1957 B70E0ACA4C                 ora a! mvi c,10! jz make0
195D CD7611             call setcdr ; may have extended the directory
                        ; Now copy entry to the directory
1960 0E00112000         mvi c,0! lxi d,fcblen! call copy$dir0
                        ; and restore the file write flag
1968 E1F177             pop h! pop a! mov m,a
                        ; and set the fcb write flag to '1'
196B CDC82C             call fix$hash
196E C36211             jmp setfwf

                open$reel:
                        ; Close the current extent, and open the next one
                        ; if possible.  rmf is true if in read mode

                if BANKED
1971 CDDC0D             call reset$copy$cr$only
                endif

1974 CD3F10             call getexta
1977 7E4F               mov a,m! mov c,a
1979 0CCD8614           inr c! call compext
197D CADA19             jz open$reel3
1980 E5C5               push h! push b
1982 CD3D18             call close
1985 C1E1               pop b! pop h
```

```
1987 3A9F0D3CC8         lda lret! inr a! rz
198C 3E1FA177           mvi a,maxext! ana c! mov m,a ; Incr extent field
                        ; Advance to module & save
1990 23237E32A7         inx h! inx h! mov a,m! sta save$mod
1996 C2A019             jnz open$reel0 ; Jump if in same module

                        open$mod:
                                ; Extent number overflow, go to next module
1999 34                         inr m ; fcb(modnum)=++1
                                ; Module number incremented, check for overflow

199A 7EE63F                     mov a,m! ani 3fh ; Mask high order bits

199D CAC919                     jz open$r$err ; cannot overflow to zero

                                ; otherwise, ok to continue with new module
                        open$reel0:
19A0 CD0319                     call set$xdcnt ; Reset xdcnt for make

                if MPM
                                call set$sdcnt
                endif

19A3 CDF014                     call search$namlen ; Next extent found?
19A6 C2B919                     jnz open$reel1
                                        ; end of file encountered
19A9 3A9C283C                           lda rmf! inr a ; 0ffh becomes 00 if read
19AD CAC919                             jz open$r$err ; sets lret = 1
                                        ; Try to extend the current file
19B0 CD1619                             call make
                                        ; cannot be end of directory
19B3 CAC919                             jz open$r$err ; with lret = 1

                if MPM
                                        call fix$olist$item
                                        call set$fcb$cks$flag
                endif

19B6 C3BC19                             jmp open$reel2
                                open$reel1:
                                        ; not end of file, open
19B9 CDDE17                             call open$copy

                if MPM
                                        call set$fcb$cks$flag
                endif

                                open$reel2:

                if not MPM
19BC CD4112                             call set$lsn
                endif

19BF CD5810                             call getfcb ; Set parameters
19C2 AF32A828C3                         xra a! sta vrecord! jmp sta$ret ; lret = 0
                                        ; ret ; with lret = 0
                                open$r$err:
                                        ; Restore module and extent
19C9 CD4B113AA7                         call getmodnum! lda save$mod! mov m,a
19D0 2B2B7E3DE6                         dcx h! dcx h! mov a,m! dcr a! ani 1fh
19D6 77C3760A                           mov m,a! jmp setlret1 ; lret = 1

                                open$reel3:
19DA 34                                 inr m ; fcb(ex) = fcb(ex) + 1
19DB CD97144F                           call get$dir$ext! mov c,a
                                        ; Is new extent beyond dir$ext?
19DF BED2EC19                           cmp m! jnc open$reel4 ; no
19E3 35                                 dcr m ; fcb(ex) = fcb(ex) - 1
                                        ; Is this a read fx?
19E4 3A9C283CCA                         lda rmf! inr a! jz set$lret1 ; yes - Don't advance ext
19EB 34                                 inr m ; fcb(ex) = fcb(ex) + 1
                                open$reel4:
19EC CD2D18                             call restore$rc
19EF CDFD17C3BC                         call set$rc! jmp open$reel2

                        seqdiskread:
                        diskread:       ; (may enter from seqdiskread)
19F5 CDEE12                     call tst$inv$fcb ; Check for valid fcb
19F8 3EFF329C28                 mvi a,true! sta rmf ; read mode flag = true (open$reel)

                        if MPM
                                sta dont$close
                        endif

                                ; Read the next record from the current fcb
19FD CD5810                     call getfcb ; sets parameters for the read
                        diskread0:
1A00 3AA82821A5                 lda vrecord! lxi h,rcount! cmp m ; vrecord-rcount
                                ; Skip if rcount > vrecord
1A07 DA191A                     jc recordok

                        if MPM
                                        call test$disk$fcb! jnz diskread0
                                        lda vrecord
                        endif

                                        ; not enough records in the extent
                                        ; record count must be 128 to continue
1A0A FE80                               cpi 128 ; vrecord = 128?
1A0C C2760A                             jnz setlret1 ; Skip if vrecord<>128
1A0F CD7119                             call open$reel ; Go to next extent if so
                                        ; Check for open ok
1A12 3A9F0DB7C2                         lda lret! ora a! jnz setlret1 ; Stop at eof
                                recordok:
                                        ; Arrive with fcb addressing a record to read

                        if BANKED
1A19 CDD50D                             call set$copy$cr$only
                        endif

1A1C CDE00F                             call index ; Z flag set if arecord = 0
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
               if MPM
                              jnz recordok1
                              call test$disk$fcb! jnz diskread0
               endif

1A1F CA760A                   jz setlret1 ; Reading unwritten data
                       recordok1:
                              ; Record has been allocated, read it
1A22 CDF20F                   call atran ; arecord now a disk address
1A25 CD4D1A                   call check$nprs
1A28 DA8210                   jc setfcb
1A2B C23F1A                   jnz read$deblock

1A2E CDB612                   call setdata
1A31 CD0C0F                   call seek ; to proper track,sector

               if BANKED
1A34 3E01CD54FF               mvi a,1! call setbnkf
               endif

1A39 CD930E                   call rdbuff ; to dma address
1A3C C38210                   jmp setfcb ; Replace parameter

               read$deblock:
1A3F 2100002296        lxi h,0! shld last$block
1A45 3E01CDFF28        mvi a,1! call deblock$dta
1A4A C38210            jmp setfcb

               check$nprs:
                       ;
                       ; on exit,  c flg          -> no i/o operation
                       ;           z flg & ~c flg -> direct(physical) i/o operation
                       ;         ~z flg & ~c flg -> indirect(deblock) i/o operation
                       ;
                       ;         Dir$cnt contains the number of 128 byte records
                       ;         to transfer directly.  This routine sets dir$cnt
                       ;         when initiating a sequence of direct physical
                       ;         i/o operations.  Dir$cnt is decremented each
                       ;         time check$nprs is called during such a sequence.
                       ;
                       ; Is direct transfer operation in progress?
1A4D 3A932847          lda blk$off! mov b,a
1A51 3A90284FA0        lda phy$msk! mov c,a! ana b! push a
1A57 3A9828FE02        lda dir$cnt! cpi 2! jc check$npr1 ; no
                       ; yes - Decrement direct record count
1A5F 3D329828          dcr a! sta dir$cnt
                       ; Are we at a new physical record?
1A63 F137C0            pop a! stc! rnz ; no - ret with c flg set
                       ; Perform physical i/o operation
1A66 AFC9              xra a! ret ; Return with z flag set and c flag reset
               check$npr1:
                       ; Are we in mid-physical record?
1A68 F1CA721A          pop a! jz check$npr11 ; no
               check$npr1a:
                       ; Is phymsk = 0?
1A6C 79B7C8            mov a,c! ora a! rz ; yes - Don't deblock
```

```
                    check$npr1b:
                              ; Deblocking required
1A6F F601C9                   ori 1! ret ; ret with z flg reset and c flg reset
                    check$npr11:
1A72 792F57                   mov a,c! cma! mov d,a ; d = ~phy$msk
1A75 21A828                   lxi h,vrecord
                              ; Is mult$num < 2?
1A78 3A9928FE02               lda mult$num! cpi 2! jc check$npr1a ; yes
1A80 86FE80DA88               add m! cpi 80h! jc check$npr2
1A86 3E80                     mvi a,80h
                    check$npr2: ; a = min(vrecord + mult$num),80h) = x
1A88 C5                       push b ; Save low(arecord) & blkmsk, phymsk
1A89 46367F                   mov b,m! mvi m,7fh ; vrecord = 7f
1A8C C5                       push b ; Save vrecord
1A8D E5                       push h ; Save .vrecord
1A8E F5                       push a ; Save x
1A8F 3A83285F1C               lda blkmsk! mov e,a! inr e! cma! ana b! mov b,a
                              ; b = vrecord & ~blkmsk
                              ; e = blkmsk + 1
1A97 E1                       pop h ; h = x
                              ; Is this a read function?
1A98 3A9C28B7CA               lda rmf! ora a! jz check$npr21 ; no
                              ; Is rcount & ~phymsk < x?
1A9F 3AA528A2BC               lda rcount! ana d! cmp h! jc check$npr23 ; yes
                    check$npr21:
1AA7 7C                       mov a,h ; a = x
                    check$npr23:
1AA8 90                       sub b ; a = a - vrecord & ~blkmsk
1AA9 4F                       mov c,a ; c = max # of records from beginning of curr blk
                              ; Is c < blkmsk+1?
1AAA BBDA1C1B                 cmp e! jc check$npr8 ; yes

                    if BANKED
1AAE C5                       push b ; c = max # of records
                              ; Compute maximum disk map position
1AAF CDA40F                   call dm$position
1AB2 47                       mov b,a ; b = index of last block in extent
                              ; Does the last block # = the current block #?
1AB3 3A9F28B85F               lda dminx! cmp b! mov e,a! jz check$npr5 ; yes
                              ; Compute # of blocks in sequence
1ABB 4FC50600                 mov c,a! push b! mvi b,0
1ABF CDCC0F                   call get$dm ; hl = current block #
                    check$npr4:
                              ; Get next block #
1AC2 E503CDCC0F               push h! inx b! call get$dm
1AC7 D113                     pop d! inx d
                              ; Does next block # = previous block # + 1?
1AC9 7A9457                   mov a,d! sub h! mov d,a
1ACC 7B95B2CAC2               mov a,e! sub l! ora d! jz check$npr4 ; yes
                              ; Is next block # = 0?
1AD2 7CB5C2F51A               mov a,h! ora l! jnz check$npr45 ; no
                              ; Is this a read function?
1AD7 3A9C28B7C2               lda rmf! ora a! jnz check$npr45 ; no
                              ; Is next block # > maxall?
1ADE 2A85287D93               lhld maxall! mov a,l! sub e
1AE3 7C9ADAF51A               mov a,h! sbb d! jc check$npr45 ; yes
```

```
                                ; Is next block # allocated?
1AEB C5D5424B                   push b! push d! mov b,d! mov c,e
1AEC CD4E13E1C1                 call getallocbit! pop h! pop b
1AF1 1FD2C21A                   rar! jnc check$npr4 ; no - it will be later
                      check$npr45:
1AF5 0DD1                       dcr c! pop d
                                ; Is max dm position less than c?
1AF7 7AB9DAFD1A                 mov a,d! cmp c! jc check$npr5 ; yes
1AFC 79                         mov a,c ; no
                      check$npr5: ; a = index of last block
1AFD 934704                     sub e! mov b,a! inr b ; b = # of consecutive blks
1B00 3A832B3C4F                 lda blkmsk! inr a! mov c,a
                      check$npr6:
1B05 05CA0D1B                   dcr b! jz check$npr7
1B09 81C3051B                   add c! jmp check$npr6
                      check$npr7:
1B0D C1                         pop b
1B0E 41                         mov b,c ; b = max # of records
1B0F 4F                         mov c,a ; c = (# of consecutive blks)*(blkmsk+1)
1B10 3A9C2BB7CA                 lda rmf! ora a! jz check$npr8
1B17 78B9DA1D1B                 mov a,b! cmp c! jc check$npr9
                      else
                                mov c,e ; multis-sector max = 1 block in non-banked systems
                      endif

                      check$npr8:
1B1C 79                         mov a,c
                      check$npr9:
                                ; Restore vrecord
1B1D E1C170                     pop h! pop b! mov m,b
1B20 C1                         pop b
                                ; a = max # of consecutive records including current blk
                                ; b = low(arecord) & blkmsk
                                ; c = phymsk
                                ; Is mult$num > a - b
1B21 21992B56                   lxi h,mult$num! mov d,m
1B25 90BAD22B1B                 sub b! cmp d! jnc check$npr10
1B2A 57                         mov d,a ; yes - use smaller value to compute dir$cnt
                      check$npr10:
                                ; Does this operation involve at least 1 physical record?
1B2B 792FA23298                 mov a,c! cma! ana d! sta dir$cnt! jz check$npr1b ; Deblocking required
                                ; Flush any pending buffers before doing multiple reads
1B34 F53A9C2BB7                 push a! lda rmf! ora a! jz check$npr10a
1B3C CD2025CDB6                 call flushx! call setdata
                      check$npr10a:
1B42 F167                       pop a! mov h,a ; Save # of 128 byte records
                                ; Does this operation involve more than 1 physical record?
                                ; Register h contains number of 128 byte records
1B44 CD9D0F7C                   call shr$physhf! mov a,h
1B48 FE014FC445                 cpi 1! mov c,a! cnz mult$iof ; yes - Make bios call
1B4E AFC9                       xra a! ret ; Return with z flg set

                      if MPM

                      test$unlocked:
                                lda high$ext! ani 80h! ret
```

```

                test$disk$fcb:
                        call test$unlocked! rz
                        lda dont$close! ora a! rz
                        call close1
                test$disk$fcb1:
                        pop d
                        lxi h,lret! inr m! mvi a,11! jz sta$ret
                        mvi m,0
                        push d
                        call getrcnta! mov a,m! sta rcount ; Reset rcount
                        xra a! sta dont$close
                        inr a! ret
                endif

                reset$fwf:
1B50 CD4B11             call getmodnum ; hl=.fcb(modnum), a=fcb(modnum)
                        ; Reset the file write flag to mark as written fcb
1B53 E67F               ani (not fwfmsk) and 0ffh ; bit reset
1B55 77                 mov m,a ; fcb(modnum) = fcb(modnum) and 7fh
1B56 C9                 ret

                set$filewf:
1B57 CD4B11E640         call getmodnum! ani 0100$0000b! push a
1B5D 7EF64077F1         mov a,m! ori 0100$0000b! mov m,a! pop a! ret

                seqdiskwrite:
                diskwrite:      ; (may enter here from seqdiskwrite above)
1B63 3E00329C28         mvi a,false! sta rmf ; read mode flag
                        ; Write record to currently selected file

1B68 CD3911             call check$write ; in case write protected

                if BANKED
1B6B 3AB628B7           lda xfcb$read$only! ora a
1B6F 3E03C2E523         mvi a,3! jnz set$aret
                endif

1B74 3AB528             lda high$ext

                if MPM
                        ani 0100$0000b
                else
1B77 B7                 ora a
                endif

                        ; Z flag reset if r/o mode
1B78 3E03C2E523         mvi a,3! jnz set$aret

1B7D 2ADBFB             lhld info ; hl = .fcb(0)
1B80 CD2B11             call check$rofile ; may be a read-only file

1B83 CDEE12             call tst$inv$fcb ; Test for invalid fcb

1B86 CD5321             call update$stamp

```

```
1B89 CD5810           call getfcb ; to set local parameters
1B8C 3AA828FEB0       lda vrecord! cpi lstrec+1 ; vrecord-128
1B91 DA9C1B           jc diskwrite0
1B94 CD7119           call open$reel ; vrecord = 128, try to open next extent
1B97 3A9F0DB7C0       lda lret! ora a! rnz ; no available fcb
                disk$write0:

                if MPM
                      mvi a,0ffh! sta dont$close
                disk$write1:

                endif

                      ; Can write the next record, so continue
1B9C CDE00F           call index ; Z flag set if arecord = 0
1B9F CABA1B           jz diskwrite2
                      ; Was the last write operation for the same block & drive?
1BA2 21AA281195       lxi h,adrive! lxi d,last$drive! mvi c,3
1BAA CD1F0ECAB5       call compare! jz diskwrite15 ; yes
                      ; no - force preread in blocking/deblocking
1BB0 3EFF329428       mvi a,0ffh! sta last$off
                diskwrite15:

                if MPM
                      ; If file is unlocked, verify record is not locked
                      ; Record has to be allocated to be locked
                      call test$unlocked! jz not$unlocked
                      call atran! mov c,a
                      lda mult$cnt! mov b,a! push b
                      call test$lock! pop b
                      xra a! mov c,a! push b
                      jmp diskwr10
                not$unlocked:
                      inr a
                endif

1BB5 0E00             mvi c,0 ; Marked as normal write operation for wrbuff
1BB7 C3111C           jmp diskwr1
                diskwrite2:

                if MPM
                            call test$disk$fcb! jnz diskwrite1
                endif

                if BANKED
1BBA CDDC0D                 call reset$copy$cr$only
                endif

                            ; not allocated
                            ; The argument to getblock is the starting
                            ; position for the disk search, and should be
                            ; the last allocated block for this file, or
                            ; the value 0 if no space has been allocated
1BBD CDA40F                 call dm$position
1BC0 329F28                 sta dminx ; Save for later
1BC3 010000                 lxi b,0000h ; May use block zero
```

```
1BC6 87CAD11B                 ora a! jz nopblock ; Skip if no previous block
                                      ; Previous block exists at a
1BCA 4F0B                             mov c,a! dcx b ; Previous block # in bc
1BCC CDCC0F                           call getdm ; Previous block # to hl
1BCF 444D                             mov b,h! mov c,l ; bc=prev block#
                              nopblock:
                                      ; bc = 0000, or previous block #
1BD1 CDB816                           call get$block ; block # to hl
                              ; Arrive here with block# or zero
1BD4 7DB4C2DE1B               mov a,l! ora h! jnz blockok
                                      ; Cannot find a block to allocate
1BD9 3E02C3720A                       mvi a,2! jmp sta$ret ; lret=2
                              blockok:

               if MPM
                              call set$fcb$cks$flag
               endif

                              ; allocated block number is in hl
1BDE 22AB282296               shld arecord! shld last$block! xra a! sta last$off
1BE8 3AAA283295               lda adrive! sta lastdrive
1BEE EB                       xchg ; block number to de
1BEF 2ADBFB0110               lhld info! lxi b,dskmap! dad b ; hl=.fcb(dskmap)
1BF6 3AA328B7                 lda single! ora a ; Set flags for single byte dm
1BFA 3A9F28                   lda dminx ; Recall dm index
1BFD CA071C                   jz allocwd ; Skip if allocating word
                                      ; Allocating a byte value
1C00 CD461173                         call addh! mov m,e ; single byte alloc
1C04 C30F1C                           jmp diskwru ; to continue
                              allocwd:
                              ; Allocate a word value
1C07 4F0600                           mov c,a! mvi b,0 ; double(dminx)
1C0A 0909                             dad b! dad b ; hl=.fcb(dminx*2)
1C0C 732372                           mov m,e! inx h! mov m,d ; double wd
                              diskwru:
                              ; disk write to previously unallocated block
1C0F 0E02                     mvi c,2 ; marked as unallocated write
                      diskwr1:
                      ; Continue the write operation of no allocation error
                      ; c = 0 if normal write, 2 if to prev unalloc block
1C11 C5               push b ; Save write flag
1C12 CDF20F           call atran ; arecord set
               diskwr10:
1C15 3ADFFBFE28               lda fx! cpi 40! jnz diskwr11 ; fx ^= wrt rndm zero fill
1C1D 793D3DC284               mov a,c! dcr a! dcr a! jnz diskwr11 ; old allocation

                              ; write random zero fill + new block

1C23 C1F5                     pop b! push a ; zero write flag
1C25 2AAB28E5                 lhld arecord! push h
1C29 2190285E1C               lxi h,phymsk! mov e,m! inr e! mov d,a! push d
1C30 2A7928                   lhld dirbcba

               if BANKED
1C33 5E2356EB                 mov e,m! inx h! mov d,m! xchg
               fill00:
```

```
1C37 E5CD782BD1                push h! call get$next$bcba! pop d! jnz fill00
1C3F EB                        xchg
               endif

                               ; Force prereads in blocking/deblocking
                               ; Discard BCB
1C40 3D32942877                dcr a! sta last$off! mov m,a
1C45 CDBC12                    call setdir1 ; Set dma to BCB buffer
                               ; Zero out BCB buffer
1C48 D1D5AF                    pop d! push d! xra a
                      fill0:
1C4B 772314F24B                mov m,a! inx h! inr d! jp fill0
1C51 571DC24B1C                mov d,a! dcr e! jnz fill0
                               ; Write 1st physical record of block
1C56 2AB1280E02                lhld arecord1! mvi c,2
                      fill1:
1C5B 22AB28C5CD                shld arecord! push b! call discard$data$bcb
1C62 CD0C0F                    call seek

               if BANKED
1C65 AFCD54FF                  xra a! call setbnkf
               endif

1C69 C1CD9E0E                  pop b! call wrbuff
1C6D 2AAB28D1D5                lhld arecord! pop d! push d
                               ; Continue writing until blkmsk & arecord = 0
1C72 193A8328A5                dad d! lda blkmsk! ana l! mvi c,0! jnz fill1
                               ; Restore arecord
1C7C E1E122AB28                pop h! pop h! shld arecord

1C81 CDB612                    call setdata ; Restore dma
                      diskwr11:

1C84 D13AA82857       pop d! lda vrecord! mov d,a ; Load and save vrecord
1C89 D5CD4D1A         push d! call check$nprs

1C8D DABA1C           jc dont$write
1C90 CA9B1C           jz write

1C93 3E02             mvi a,2 ; deblock write code
1C95 CDFF28           call deblock$dta
1C98 C3BA1C           jmp dont$write
               write:
1C9B CDB612           call setdata
1C9E CD0C0F           call seek

               if BANKED
1CA1 3E01CD54FF       mvi a,1! call setbnkf
               endif

                      ; Discard matching BCB if write is direct
1CA6 CD4A12           call discard$data$bcb

                      ; Set write flag to zero if arecord & blkmsk ^= 0

1CA9 C1C53AAB28       pop b! push b! lda arecord
```

```
1CAE 218328A6CA         lxi h,blkmsk! ana m! jz write0
1CB5 0E00               mvi c,0
                  write0:
1CB7 CD9E0E             call wrbuff

                  dont$write:
1CBA C1                 pop b ; c = 2 if a new block was allocated, 0 if not
                        ; Increment record count if rcount<=vrecord
1CBB 7821A528BE         mov a,b! lxi h,rcount! cmp m ; vrecord-rcount
1CC0 DAC71C             jc diskwr2
                                ; rcount <= vrecord
1CC3 7734                       mov m,a! inr m ; rcount = vrecord+1

                  if MPM
                                call test$unlocked! jz write1

                                ; for unlocked files
                                ;   rcount = rcount & (~ blkmsk) + blkmsk + 1

                                lda blkmsk! mov b,a! inr b! cma! mov c,a
                                mov a,m! dcr a! ana c! add b! mov m,a
                        write1:
                  endif

1CC5 0E02                       mvi c,2 ; Mark as record count incremented
                        diskwr2:
                        ; a has vrecord, c=2 if new block or new record#
1CC7 0D0DC2CF1C         dcr c! dcr c! jnz noupdate
1CCC CD501B                     call reset$fwf

                  if MPM
                                call test$unlocked! jz noupdate
                                lda rcount! call getrcnta! mov m,a
                                call close
                                call test$disk$fcb1
                  endif

                  noupdate:
                        ; Set file write flag if reset
1CCF CD571B             call set$filewf

                  if BANKED
1CD2 C2DE1C             jnz disk$write3
                        ; Reset fcb file write flag to ensure t3' gets
                        ; reset by the close function
1CD5 CD501B             call reset$fwf
1CD8 CDDC0D             call reset$copy$cr$only
1CDB C38210             jmp setfcb
                  disk$write3:
1CDE CDD50D             call set$copy$cr$only
                  else
                        cz reset$fwf
                  endif
1CE1 C38210             jmp setfcb ; Replace parameters
                        ; ret
                        rseek:
                                ; Random access seek operation, c=0ffh if read mode
                                ; fcb is assumed to address an active file control block
                                ; (1st block of FCB = 0ffffh if previous bad seek)
1CE4 C5                         push b ; Save r/w flag
1CE5 2ADBFBEB                   lhld info! xchg ; de will hold base of fcb
1CE9 21210019                           lxi h,ranrec! dad d ; hl=.fcb(ranrec)
1CED 7EE67FF5                           mov a,m! ani 7fh! push psw ; record number
1CF1 7E17                               mov a,m! ral ; cy=lsb of extent#
1CF3 237E17E61F                         inx h! mov a,m! ral! ani 11111b ; a=ext#
1CF8 4F                                 mov c,a ; c holds extent number, record stacked

1CF9 7EE6F023B6                         mov a,m! ani 1111$0000b! inx h! ora m
1CFE 0F0F0F0F47                         rrc! rrc! rrc! rrc! mov b,a
                                        ; b holds module #

                                        ; Check high byte of ran rec <= 3
1D03 7E                                 mov a,m
1D04 E6FCE12E06                         ani 1111$1100b! pop h! mvi l,6! mov a,h

                                        ; Produce error 6, seek past physical end
1D0A C2B01D                             jnz seekerr

                                        ; otherwise, high byte = 0, a = sought record
1D0D 21200019                           lxi h,nxtrec! dad d ; hl = .fcb(nxtrec)
1D11 77                                 mov m,a ; sought rec# stored away

                                ; Arrive here with b=mod#, c=ext#, de=.fcb, rec stored
                                ; the r/w flag is still stacked.  compare fcb values

1D12 3ADFFBFE63                         lda fx! cpi 99! jz rseek3
                                        ; Check module # first
1D1A D5CDE812D1                         push d! call chk$inv$fcb! pop d! jz ranclose
1D22 210E001978                         lxi h,modnum! dad d! mov a,b ; b=seek mod#
1D27 96E63FC25B                         sub m! ani 3fh! jnz ranclose ; same?
                                        ; Module matches, check extent
1D2D 210C0019                           lxi h,extnum! dad d
1D31 7EB9CAA41D                         mov a,m! cmp c! jz seekok2 ; extents equal
1D36 CD8614C25B                         call compext! jnz ranclose
                                        ; Extent is in same directory fcb
1D3C C5CD9714C1                         push b! call get$dir$ext! pop b
1D41 B9D2501D                           cmp c! jnc rseek2 ; jmp if dir$ext > ext
1D45 D1D51CC250                         pop d! push d! inr e! jnz rseek2 ; jmp if write fx
1D4B 1CD1C3760A                         inr e! pop d! jmp set$lret1 ; error - reading unwritten data
                                rseek2:
1D50 71                                 mov m,c ; fcb(ext) = c
1D51 4F                                 mov c,a ; c = dir$ext
                                        ; hl=.fcb(ext),c=dir ext
1D52 CD2D18                             call restore$rc
1D55 CDFD17                             call set$rc
1D58 C3A11D                             jmp seekok1
                                ranclose:
1D5B C5D5                               push b! push d ; Save seek mod#,ext#, .fcb
1D5D CD3D18                             call close ; Current extent closed
1D60 D1C1                               pop d! pop b ; Recall parameters and fill
1D62 2E03                               mvi l,3 ; Cannot close error #3
```

```
1D64 3A9F0D3CCA                 lda lret! inr a! jz seekerr
                        rseek3:
1D6B CD0319                     call set$xdcnt ; Reset xdcnt for make

                if MPM
                                call set$sdcnt
                endif

1D6E 210C0019E5                 lxi h,extnum! dad d! push h
1D73 5671                       mov d,m! mov m,c ; fcb(extnum)=ext#
1D75 23237E5FD5                 inx h! inx h! mov a,m! mov e,a! push d
1D7A E6408077                   ani 040h! ora b! mov m,a
                                ; fcb(modnum)=mod#
1D7E CDDA17                     call open ; Is the file present?
1D81 3A9F0D3CC2                 lda lret! inr a! jnz seekok ; Open successful?
                                ; Cannot open the file, read mode?
1D88 D1E1C1                     pop d! pop h! pop b ; r/w flag to c (=0ffh if read)
1D8B C5E5D5                     push b! push h! push d ; Restore stack
1D8E 2E04                       mvi l,4 ; Seek to unwritten extent #4
1D90 0C                         inr c ; becomes 00 if read operation
1D91 CAA91D                     jz badseek ; Skip to error if read operation
                                ; Write operation, make new extent
1D94 CD1619                     call make
1D97 2E05                       mvi l,5 ; cannot create new extent #5
1D99 CAA91D                     jz badseek ; no dir space

                if MPM
                                call fix$olist$item
                endif

                                ; file make operation successful
                        seekok:
1D9C C1C1                       pop b! pop b ; Discard top 2 stacked items

                if MPM
                                call set$fcb$cks$flag
                else
1D9E CD4112                     call set$lsn
                endif

                        seekok1:

                if BANKED
1DA1 CDDC0D                     call reset$copy$cr$only
                endif

                        seekok2:
1DA4 C1                         pop b ; Discard r/w flag or .fcb(ext)
1DA5 AFC3720A                   xra a! jmp sta$ret ; with zero set
                        badseek:
                                ; Restore fcb(ext) & fcb(mod)
1DA9 D1E3                       pop d! xthl ; Save error flag
1DAB 72232373                   mov m,d! inx h! inx h! mov m,e
1DAF E1                         pop h ; Restore error flag
                        seekerr:
```

```
                if BANKED
1DB0 CDDC0D                     call reset$copy$cr$only ; Z flag set
1DB3 3C                         inr a ; Reset Z flag
                endif

1DB4 C1                         pop b ; Discard r/w flag
1DB5 7DC3720A                   mov a,l! jmp sta$ret ; lret=#, nonzero

                randiskread:
                        ; Random disk read operation
1DB9 0EFF               mvi c,true ; marked as read operation
1DBB CDE41C             call rseek
1DBE CCF519             cz diskread ; if seek successful
1DC1 C9                 ret

                randiskwrite:
                        ; Random disk write operation
1DC2 0E00               mvi c,false ; marked as write operation
1DC4 CDE41C             call rseek
1DC7 CC631B             cz diskwrite ; if seek successful
1DCA C9                 ret

                compute$rr:
                        ; Compute random record position for getfilesize/setrandom
1DCB EB19               xchg! dad d
                        ; de=.buf(dptr) or .fcb(0), hl = .f(nxtrec/reccnt)
1DCD 4E0600             mov c,m! mvi b,0 ; bc = 0000 0000 ?rrr rrrr
1DD0 210C00197E         lxi h,extnum! dad d! mov a,m! rrc! ani 80h ; a=e000 0000
1DD8 814F3E0088         add c! mov c,a! mvi a,0! adc b! mov b,a
                        ; bc = 0000 000? errrr rrrr
1DDE 7E0FE60F80         mov a,m! rrc! ani 0fh! add b! mov b,a
                        ; bc = 000? eeee errrr rrrr
1DE4 210E00197E         lxi h,modnum! dad d! mov a,m ; a=xxmm mmmm
1DE9 87878787           add a! add a! add a! add a ; cy=m a=mmmm 0000

1DED B78047F5           ora a! add b! mov b,a! push psw ; Save carry
1DF1 7E1F1F1F1F         mov a,m! rar! rar! rar! rar! ani 0000$0011b ; a=0000 00mm
1DF8 6FF13E008D         mov l,a! pop psw! mvi a,0! adc l ; Add carry
1DFD C9                 ret

                compare$rr:
1DFE 5F                 mov e,a ; Save cy
1DFF 79965723           mov a,c! sub m! mov d,a! inx h ; 1st byte
1E03 789E23             mov a,b! sbb m! inx h ; middle byte
1E06 F5B257F1           push a! ora d! mov d,a! pop a
1E0A 7B9E               mov a,e! sbb m ; carry if .fcb(ranrec) > directory
1E0C C9                 ret

                set$rr:
1E0D 732B702B71         mov m,e! dcx h! mov m,b! dcx h! mov m,c! ret

                getfilesize:
                        ; Compute logical file size for current fcb
                        ; Zero the receiving ranrec field
1E13 CD3710E5           call get$rra! push h ; Save position
1E17 7223722372         mov m,d! inx h! mov m,d! inx h! mov m,d ; =00 00 00
```

```
1E1C CDF514            call search$extnum
                       getsize:
1E1F CA3E1E                    jz setsize
                               ; current fcb addressed by dptr
1E22 CD4011110F                call getdptra! lxi d,reccnt ; ready for compute size
1E28 CDCB1D                    call compute$rr
                               ; a=0000 00mm bc = mmmm eeee errr rrrr
                               ; Compare with memory, larger?
1E2B E1E5                      pop h! push h ; Recall, replace .fcb(ranrec)
1E2D CDFE1DD40D                call compare$rr! cnc set$rr
1E33 CD0315                    call searchn
1E36 3E00329F0D                mvi a,0! sta aret
1E3B C31F1E                    jmp getsize
                       setsize:

1E3E E1                pop h ; Discard .fcb(ranrec)
1E3F C9                ret

                setrandom:
                       ; Set random record from the current file control block
1E40 EB112000          xchg! lxi d,nxtrec ; Ready params for computesize
1E44 CDCB1D            call compute$rr ; de=info, a=0000 00mm, bc=mmmm eeee errr rrrr
1E47 21210019          lxi h,ranrec! dad d ; hl = .fcb(ranrec)
1E4B 7123702377        mov m,c! inx h! mov m,b! inx h! mov m,a ; to ranrec
1E50 C9                ret

                disk$select:
                       ; Select disk info for subsequent input or output ops
1E51 32AA28            sta adrive
                disk$select1: ; called by deblock
1E54 77                mov m,a ; curdsk = seldsk or adrive
1E55 57                mov d,a ; Save seldsk in register D for selectdisk call
1E56 2A6528CD1D        lhld dlog! call test$vector ; test$vector does not modify DE
1E5C 5FD5              mov e,a! push d ; Send to seldsk, save for test below
1E5E CD330EE1          call selectdisk! pop h ; Recall dlog vector
1E62 D2AB0D            jnc sel$error ; returns with C flag set if select ok
                       ; Is the disk logged in?
1E65 2D                dcr l ; reg l = 1 if so
1E66 C9                ret

                tmpselect:
1E67 21A42873          lxi h,seldsk! mov m,e

                curselect:
1E6B 3AA42821A9        lda seldsk! lxi h,curdsk! cmp m! jnz select
1E75 FEFFC0            cpi 0ffh! rnz ; return if seldsk ~= ffh

                select:
1E78 CD511E            call disk$select

                if MPM
                       jnz select1 ; no
                       ; yes - drive previously logged in
                       lhld rlog! call test$vector
                       sta rem$drv! ret ; Set rem$drv & return
                select1:
```

```

               else
1E7B C8               rz ; yes - drive previously logged in
               endif

1E7C CDEB13           call initialize ; Log in the directory

                      ; Increment login sequence # if odd
1E7F 2A71287EE6       lhld lsn$add! mov a,m! ani 1! push a! add m! mov m,a
1E88 F1C4E811         pop a! cnz set$rlog

1E8C CD0511           call set$dlog

               if MPM
                      lxi h,chksiz+1! mov a,m! ral! mvi a,0! jc select2
                      lxi d,rlog! call set$cdisk ; rlog=set$cdisk(rlog)
                      mvi a,1
               select2:
                      sta rem$drv
               endif

1E8F C9               ret

               reselectx:
1E90 AF32B528         xra a! sta high$ext

               if BANKED
1E94 32B628           sta xfcb$read$only
               endif

1E97 C3B91E           jmp reselect1

               reselect:
                      ; Check current fcb to see if reselection necessary
1E9A 017F80           lxi b,807fh
1E9D 2ADBFB1107       lhld info! lxi d,7! xchg! dad d

               if BANKED
                      ; xfcb$read$only = 80h & fcb(7)
1EA5 7EA032B628       mov a,m! ana b! sta xfcb$read$only
                      ; fcb(7) = fcb(7) & 7fh
1EAA 7EA177           mov a,m! ana c! mov m,a
               endif

               if MPM
                      ; if fcb(8) & 80h
                      ;    then fcb(8) = fcb(8) & 7fh, high$ext = 60h
                      ;    else high$ext = fcb(ext) & 0e0h
                      inx h! lxi d,4
                      mov a,m! ana c! cmp m! mov m,a! mvi a,60h! jnz reselect0
                      dad d! mvi a,0e0h! ana m
               reselect0:
                      sta high$ext
               else
                      ; high$ext = 80h & fcb(8)
1EAD 237EA032B5       inx h! mov a,m! ana b! sta high$ext
```

```
                      ; fcb(8) = fcb(8) & 7fh
1EB3 7EA177           mov a,m! ana c! mov m,a
               endif

                      ; fcb(ext) = fcb(ext) & 1fh
1EB6 CD5A11           call clr$ext
               reselect1:

1EB9 210000           lxi h,0

               if BANKED
1EBC 2215FD           shld make$xfcb ; make$xfcb,find$xfcb = 0
               endif
1EBF 2217FD           shld xdcnt ; required by directory hashing

1EC2 AF32BD28         xra a! sta search$user0
1EC6 3D32DDFB         dcr a! sta resel ; Mark possible reselect
1ECA 2ADBFB7E         lhld info! mov a,m ; drive select code
1ECE 32BF28           sta fcbdsk ; save drive code
1ED1 E61F             ani 1$1111b ; non zero is auto drive select
1ED3 3D               dcr a ; Drive code normalized to 0..30, or 255
1ED4 329E28           sta linfo ; Save drive code
1ED7 FEFFCADF1E       cpi 0ffh! jz noselect
                              ; auto select function, seldsk saved above
1EDC 32A428                   sta seldsk
                      noselect:
1EDF CD6B1E                   call curselect
                              ; Set user code
1EE2 3AE0FB                   lda usrcode ; 0...15
1EE5 2ADBFB77                 lhld info! mov m,a
                      noselect0:
                              ; Discard directory BCB's if drive is removable
                              ; and fx = 15,17,19,22,23,30 etc.
1EE9 CDF511CC58               call tst$log$fxs! cz discard$dir
                              ; Check for media change on currently slected disk
1EEF CDF51E                   call check$media
                              ; Check for media change on any other disks
1EF2 C32F1F                   jmp check$all$media

               check$media:
                      ; Check media if DPH media flag set.
                      ; Is DPH media flag set?
1EF5 CD0C12C8         call test$media$flag! rz ; no
                      ; Test for media change by reading directory
                      ; to current high water mark or until media change
                      ; is detected.
                      ; First reset DPH media flag & discard directory BCB's
1EF9 3600             mvi m,0
1EFB CD5812           call discard$dir
1EFE 2AE1FBE5         lhld dcnt! push h
1F02 CD800ECD01       call home! call set$end$dir
               check$media1:
1F08 0E00CD0E13       mvi c,false! call r$dir
1F0D 21DEFB7EB7       lxi h,relog! mov a,m! ora a! jz check$media2
1F15 3600E13ADF       mvi m,0! pop h! lda fx! cpi 48! rz
1F1E CD3312C313       call drv$relog! jmp chk$exit$fxs
```

```
                    check$media2:
1F24 CD6911DA08              call comp$cdr! jc check$media1
1F2A E122E1FBC9              pop h! shld dcnt! ret

                    check$all$media:
                             ; This routine checks all logged-in drives for
                             ; a set DPH media flag and pending buffers.  It reads
                             ; the directory for these drives to verify that media
                             ; has not changed.  If media has changed, the drives
                             ; get reset (but not relogged-in).
                             ; Is SCB media flag set?
1F2F 21F0FB7EB7              lxi h,media$flag! mov a,m! ora a! rz ; no
                             ; Reset SCB media flag
1F35 3600                    mvi m,0
                             ; Test logged-in drives only
1F37 2A652B3E10              lhld dlog! mvi a,16
                    chk$am1:
1F3C 3D29D2511F              dcr a! dad h! jnc chk$am2
                             ; A = drive #
                             ; Select drive
1F41 F5E521A928              push a! push h! lxi h,curdsk! call disk$select
                             ; Does drive have pending data buffers?
1F49 CD581FC4F5              call test$pending! cnz check$media ; yes
1F4F E1F1                    pop h! pop a
                    chk$am2:
1F51 B7C23C1F                ora a! jnz chk$am1
1F55 C36B1E                  jmp curselect

                    test$pending:
                             ; On return, Z flag reset if buffer pending

                             ; Does dta$bcba = 0ffffh
1F58 2A7B287DA4              lhld dta$bcba! mov a,l! ana h! inr a! rz ; yes

                    if BANKED

                    test$p1:
                             ; Does bcb addr = 0?
1F5F 5E2356                  mov e,m! inx h! mov d,m
1F62 7BB2C8                  mov a,e! ora d! rz ; yes - no pending buffers
1F65 210400                  lxi h,4
                    else
                             lxi d,4
                    endif

                             ; Is buffer pending?
1F68 197EB7                  dad d! mov a,m! ora a ; A ^= 0 if so

                    if BANKED
1F6B C0                      rnz ; yes
                             ; no - advance to next bcb
1F6C 210D0019C3              lxi h,13! dad d! jmp test$p1
                    else
                             ret
                    endif

```

```
                     get$dir$mode:
1F73 2A6F287E               lhld drvlbla! mov a,m

                     if not BANKED
                            ani 7fh ; Mask off password bit
                     endif

1F77 C9                     ret

                     if BANKED

                     chk$password:
1F78 CD731FE680             call get$dir$mode! ani 80h! rz

                     chk$pw:                  ; Check password
1F7E CD8320C8               call get$xfcb! rz ; a = xfcb options
1F82 C30220                 jmp cmp$pw

                     chk$pw$error:
                            ; Disable special searches
1F85 AF3218FD               xra a! sta xdcnt+1
                            ; pw$fcb = dir$xfcb
1F89 CD4011EB               call getdptra! xchg
1F8D 0E0C21D728             mvi c,12! lxi h,pw$fcb! push h
1F93 CD290E1A23             call move! ldax d! inx h! mov m,a! pop d
1F9A 2ADBFB7E12             lhld info! mov a,m! stax d
                            ; push original info and xfcb password mode
                            ; info = .pw$fcb
1F9F E5EB22DBFB             push h! xchg! shld info
                            ; Does fcb(ext = 0, mod = 0) exist?
1FA4 CDF014CAE4             call search$namlen! jz chk$pwe2 ; no
                            ; Does sfcb exist for fcb ?
1FAA CD1021B7C2             call get$dtba$8! ora a! jnz chk$pwe1 ; no
1FB1 EB21E428               xchg! lxi h,pw$mode
                            ; Is sfcb password mode nonzero?
1FB5 461A77B7CA             mov b,m! ldax d! mov m,a! ora a! jz chk$pwe2 ; no
                            ; Do password modes match?
1FBC A8E6E0CAD2             xra b! ani 0e0h! jz chk$pwe1 ; yes
                            ; no - update xfcb to match sfcb
1FC2 CD8320CAD2             call get$xfcb! jz chk$pwe1 ; no xfcb (error)
1FC8 3AE42877CD             lda pw$mode! mov m,a! call nowrite! cz seek$copy
                     chk$pwe1:
1FD2 E122DBFB               pop h! shld info
1FD6 3ADFFBFE0F             lda fx! cpi 15! rz! cpi 22! rz

                     pw$error:       ; password error
1FDF 3E07C3E523             mvi a,7! jmp set$aret

                     chk$pwe2:
1FE4 AF32E428               xra a! sta pw$mode
1FE8 CD1A11C2FD             call nowrite! jnz chk$pwe3
                            ; Delete xfcb
1FEE CD8320F5               call get$xfcb! push a
1FF2 2ADBFB7EF6             lhld info! mov a,m! ori 10h! mov m,a
1FF9 F1C48716               pop a! cnz delete$10
                     chk$pwe3:
                        ; Restore info
1FFD E122DBFBC9         pop h! shld info! ret

                cmp$pw:                 ; Compare passwords
2002 2346               inx h! mov b,m
2004 78B7C21D20         mov a,b! ora a! jnz cmp$pw2
2009 545D2323           mov d,h! mov e,l! inx h! inx h
200D 0E09               mvi c,9
                cmp$pw1:
200F 237E0DC8           inx h! mov a,m! dcr c! rz
2013 B7CA0F20           ora a! jz cmp$pw1
2017 FE20CA0F20         cpi 20h! jz cmp$pw1
201C EB                 xchg
                cmp$pw2:
201D 110A0019EB         lxi d,(23-ubytes)! dad d! xchg
2022 2A19FD0E08         lhld xdmaad! mvi c,8
                cmp$pw3:
2027 1AA8BEC234         ldax d! xra b! cmp m! jnz cmp$pw4
202D 1B230DC227         dcx d! inx h! dcr c! jnz cmp$pw3
2033 C9                 ret
                cmp$pw4:
2034 1B0DC23420         dcx d! dcr c! jnz cmp$pw4
2039 13                 inx d

                if MPM
                        call get$df$pwa! inr a! jnz cmp$pw5
                        inr a! ret
                cmp$pw5:

                else
203A 21E728             lxi h,df$password
                endif

203D 0E08C31F0E         mvi c,8! jmp compare

                if MPM

                get$df$pwa:    ; a = ff => no df pwa
                        call rlr! lxi b,console! dad b
                        mov a,m! cpi 16! mvi a,0ffh! rnc
                        mov a,m! add a! add a! add a
                        mvi h,0! mov l,a! lxi b,dfpassword! dad b
                        ret
                endif

                set$pw:                 ; Set password in xfcb
2042 E5                 push h ; Save .xfcb(ex)
2043 010800             lxi b,8 ; b = 0, c = 8
2046 11080019           lxi d,(23-extnum)! dad d
204A EB2A19FD           xchg! lhld xdmaad
                set$pw0:
204E AFF5               xra a! push a
                set$pw1:
2050 7E12B7CA5E         mov a,m! stax d! ora a! jz set$pw2
2056 FE20CA5E20         cpi 20h! jz set$pw2
205B 3333F5             inx sp! inx sp! push a
```

```
                        set$pw2:
205E 8047                          add b! mov b,a
2060 1B230DC250                    dcx d! inx h! dcr c! jnz set$pw1
2066 F1B0E1C276                    pop a! ora b! pop h! jnz set$pw3
                                   ; is fx = 100 (directory label)?
206C 3ADFFBFE64                    lda fx! cpi 100! jz set$pw3 ; yes
2074 3600                          mvi m,0 ; zero xfcb(ex) - no password
                        set$pw3:
2076 130E08                        inx d! mvi c,8
                        set$pw4:
2079 1AA812130D                    ldax d! xra b! stax d! inx d! dcr c! jnz set$pw4
2081 23C9                          inx h! ret

                        get$xfcb:
2083 2ADBFB7EF5                    lhld info! mov a,m! push a
2088 F61077                        ori 010h! mov m,a
208B CDF5143E00                    call search$extnum! mvi a,0! sta lret
2093 2ADBFBC170                    lhld info! pop b! mov m,b! rz
                        get$xfcb1:
2099 CD4011EB                      call getdptra! xchg
209D 210C00197E                    lxi h,extnum! dad d! mov a,m! ani 0e0h! ori 1
20A6 C9                            ret

                        adjust$dmaad:
20A7 E52A19FD19                    push h! lhld xdmaad! dad d
20AC 2219FDE1C9                    shld xdmaad! pop h! ret

                        init$xfcb:
20B1 CD7611                        call setcdr ; may have extended the directory
20B4 011410                        lxi b,1014h ; b=10h, c=20
                        init$xfcb0:
                                   ; b = fcb(0) logical or mask
                                   ; c = zero count
20B7 C5                            push b
20B8 CD4011EB2A                    call getdptra! xchg! lhld info! xchg
                                   ; Zero extnum and modnum
20C0 1AB0771323                    ldax d! ora b! mov m,a! inx d! inx h
20C5 0E0BCD290E                    mvi c,11! call move! pop b! inr c
                        init$xfcb1:
20CC 0DC8                          dcr c! rz
20CE 360023C3CC                    mvi m,0! inx h! jmp init$xfcb1

                        chk$xfcb$password:
20D4 CD9920                        call get$xfcb1
                        chk$xfcb$password1:
20D7 E5CD0220E1                    push h! call cmp$pw! pop h! ret

                        endif

                        stamp1:
20DD 0E00C3E420                    mvi c,0! jmp stamp3
                        stamp2:
20E2 0E04                          mvi c,4
                        stamp3:
20E4 CD1221B7C0                    call get$dtba! ora a! rnz
20E9 119A12D5                      lxi d,seek$copy! push d
```

```
                    stamp4:

                    if MPM
                            push h
                            call get$stamp$add! xchg
                            pop h
                    else
20ED 11F4FB                 lxi d,stamp
                    endif

20F0 E5D5                   push h! push d
20F2 0E00CD4EFF             mvi c,0! call timef ; does not modify hl,de
20F7 0E04CD1F0E             mvi c,4! call compare
20FC 0E04D1E1C2             mvi c,4! pop d! pop h! jnz move
2103 E1C9                   pop h! ret

                    stamp5:
2105 CD40110911             call getdptra! dad b! lxi d,func$ret! push d
210D C3ED20                 jmp stamp4

                    if BANKED

                    get$dtba$8:
2110 0E08                   mvi c,8
                    endif

                    get$dtba:
                            ; c = offset of sfcb subfield (0,4,8)
                            ; Return with a = 0 if sfcb exists

                            ; Does fcb occupy 4th item of sector?
2112 3AE1FBE603             lda dcnt! ani 3! cpi 3! rz ; yes
211A 47                     mov b,a
211B 2A67281160             lhld buffa! lxi d,96! dad d
                            ; Does sfcb reside in 4th directory item?
2122 7ED621C0               mov a,m! sui 21h! rnz ; no
                            ; hl = hl + 10*lret + 1 + c
2126 78875F8787             mov a,b! add a! mov e,a! add a! add a! add e
212C 3C815F19AF             inr a! add c! mov e,a! dad d! xra a
2131 C9                     ret

                    qstamp:
                            ; Is fcb 1st logical fcb for file?
2132 CD4021C0               call qdirfcb1! rnz ; no
                    qstamp1:
                            ; Does directory label specify requested stamp?
2136 2A6F2879A6             lhld drvlbla! mov a,c! ana m! jnz nowrite ; yes - verify drive r/w
213E 3CC9                   inr a! ret ; no - return with Z flag reset

                    qdirfcb1:
                            ; Routine to determine if fcb is 1st directory fcb
                            ; for file
                            ; Is fcb(ext) & ~extmsk & 00011111b = 0?
2140 3A8428F6E0             lda extmsk! ori 1110$0000b! cma! mov b,a
2147 CD3F107EA0             call getexta! mov a,m! ana b! rnz ; no
                            ; is fcb(mod) & 0011$1111B = 0?
```

```
214D 23237EE63F        inx h! inx h! mov a,m! ani 3fh! ret ; Z flag set if zero

               update$stamp:
                       ; Is update stamping requested on drive?
2153 0E20CD3621        mvi c,0010$0000b! call qstamp1! rnz ; no
                       ; Has file been written to since it was opened?
2159 CD4B11E640        call getmodnum! ani 40h! rnz ; yes - update stamp performed
                       ; Search for 1st dir fcb
215F CD3F104636        call getexta! mov b,m! mvi m,0! push h
2166 23234E3600        inx h! inx h! mov c,m! mvi m,0! push b
                       ; Search from beginning of directory
216C CDF014            call search$namlen
                       ; Perform update stamp if dir fcb 1 found
216F C4E220            cnz stamp2
2172 AF329F0D          xra a! sta lret
                       ; Restore fcb extent and module fields
2176 C1E1702323        pop b! pop h! mov m,b! inx h! inx h! mov m,c! ret

               if MPM

               pack$sdcnt:

               ;packed$dcnt = dblk(low 15 bits) || dcnt(low 9 bits)

               ;       if sdblk = 0 then dblk = shr(sdcnt,blkshf+2)
               ;                    else dblk = sdblk
               ;       dcnt = sdcnt & (blkmsk || '11'b)
               ;
               ;       packed$dcnt format (24 bits)
               ;
               ;       12345678 12345678 12345678
               ;       23456789 .......1 ........ sdcnt (low 9 bits)
               ;       ........ 9abcdef. 12345678 sdblk (low 15 bits)
               ;
                       lhld sdblk! mov a,h! ora l! jnz pack$sdcnt1
                       lda blkshf! adi 2! mov c,a! lhld sdcnt
                       call hlrotr
               pack$sdcnt1:
                       dad h! xchg! lxi h,sdcnt! mvi b,1
                       lda blkmsk! ral! ora b! ral! ora b
                       ana m! sta packed$dcnt
                       lda blkshf! cpi 7! jnz pack$sdcnt2
                       inx h! mov a,m! ana b! jz pack$sdcnt2
                       mov a,e! ora b! mov e,a
               pack$sdcnt2:
                       xchg! shld packed$dcnt+1
                       ret

               ; olist element = link(2) || atts(1) || dcnt(3) ||
               ;                pdaddr(2) || opncnt(2)
               ;
               ;       link = 0 -> end of list
               ;
               ;       atts - 80 - open in locked mode
               ;              40 - open in unlocked mode
               ;              20 - open in read/only mode
;               20 - deleted item
;               0n - drive code (0-f)
;
;       dcnt = packed sdcnt#sdblk
;       pdaddr = process descriptor addr
;       opncnt = # of open calls - # of close calls
;                olist item freed by close when opncnt = 0
;
; llist element = link(2) || drive(1) || arecord(3) ||
;                 pdaddr(2) || .olist$item(2)
;
;       link = 0 -> end of list
;
;       drive - 0n - drive code (0-f)
;
;       arecord = record number of locked record
;       pdaddr = process descriptor addr
;       .olist$item = address of file's olist item

search$olist:
        lxi h,open$root! jmp srch$list0
search$llist:
        lxi h,lock$root! jmp srch$list0
searchn$list:
        lhld cur$pos
srch$list0:
        shld prv$pos

; search$olist, search$llist, searchn$list conventions
;
;       b = 0 -> return next item
;       b = 1 -> search for matching drive
;       b = 3 -> search for matching dcnt
;       b = 5 -> search for matching dcnt # pdaddr
;       if found then z flag is set
;                    prv$pos -> previous list element
;                    cur$pos -> found list element
;                    hl -> found list element
;       else prv$pos -> list element to insert after
;
;       olist and llist are maintained in drive order

srch$list1:
        mov e,m! inx h! mov d,m! xchg
        mov a,l! ora h! jz srch$list3
        xra a! cmp b! jz srch$list6
        inx h! inx h!
        lxi d,curdsk! mov a,m! ani 0fh! mov c,a
        ldax d! sub c! jnz srch$list4
        mov a,b! dcr a! jz srch$list5
        mov c,b! push h
        inx d! inx h! call compare
        pop h! jz srch$list5
srch$list2:
        dcx h! dcx h
        shld prv$pos! jmp srch$list1
```

```
srch$list3:
        inr a! ret
srch$list4:
        jnc srch$list2
srch$list5:
        dcx h! dcx h
srch$list6:
        shld cur$pos! ret

delete$item: ; hl -> item to be deleted
        di
        push d! push h
        mov e,m! inx h! mov d,m
        lhld prv$pos! shld cur$pos
        ; prv$pos.link = delete$item.link
        mov m,e! inx h! mov m,d

        lhld free$root! xchg
        ; free$root = .delete$item
        pop h! shld free$root
        ; delete$item.link = previous free$root
        mov m,e! inx h! mov m,d
        pop d! ei! ret

create$item: ; hl -> new item if successful
                ; z flag set if no free items
        lhld free$root! mov a,l! ora h! rz
        push d! push h! shld cur$pos
        mov e,m! inx h! mov d,m
        ; free$root = free$root.link
        xchg! shld free$root

        lhld prv$pos
        mov e,m! inx h! mov d,m
        pop h
        ; create$item.link = prv$pos.link
        mov m,e! inx h! mov m,d! dcx h
        xchg! lhld prv$pos
        ; prv$pos.link = .create$item
        mov m,e! inx h! mov m,d! xchg
        pop d! ret

set$olist$item:
        ; a = attributes
        ; hl = olist entry address
        inx h! inx h
        mov b,a! lxi d,curdsk! ldax d! ora b
        mov m,a! inx h! inx d
        mvi c,5! call move
        xra a! mov m,a! inx h! mov m,a! ret

set$sdcnt:
        mvi a,0ffh! sta sdcnt+1! ret

tst$olist:
        mvi a,0c9h! sta chk$olist05! jmp chk$olist0
```

```
chk$olist:
        xra a! sta chk$olist05
chk$olist0:
        lxi d,dcnt! lxi h,sdcnt! mvi c,4! call move
        call pack$sdcnt! mvi b,3! call search$olist! rnz
        pop d ; pop return address
        inx h! inx h
        mov a,m! ani 80h! jz openx06
        dcx h! dcx h
        push d! push h
        call compare$pds! pop h! pop d! jnz openx06
        push d ; Restore return address
chk$olist05:
        nop ; tst$olist changes this instr to ret
        call delete$item! lda pdcnt
chk$olist1:
        adi 16! jz chk$olist1
        sta pdcnt

        push a! call rlr
        lxi b,pdcnt$off! dad b! pop a
        mov m,a! ret

remove$files:  ; bc = pdaddr
        lhld cur$pos! push h
        lhld prv$pos! push h
        mov d,b! mov e,c! lxi h,open$root! shld cur$pos
remove$file1:
        mvi b,0! push d! call searchn$list! pop d! jnz remove$file2
        lxi b,6! call tst$tbl$lmt! jnz remove$file1
        inx h! inx h! mov a,m! ori 10h! mov m,a
        sta deleted$files
        jmp remove$file1
remove$file2:
        pop h! shld prv$pos
        pop h! shld cur$pos
        ret

delete$files:
        lxi h,open$root! shld cur$pos
delete$file1:
        mvi b,0! call search$nlist! rnz
        inx h! inx h! mov a,m! ani 10h! jz delete$file1
        dcx h! dcx h! call remove$locks! call delete$item
        jmp delete$file1

flush$files:
        lxi h,flushed! mov a,m! ora a! rnz
        inr m
flush$file0:
        lxi h,open$root! shld cur$pos
flush$file1:
        mvi b,1! call searchn$list! rnz
        push h! call remove$locks! call delete$item! pop h
        lxi d,6! dad d! mov e,m! inx h! mov d,m
        lxi h,pdcnt$off! dad d! mov a,m! ani 1! jnz flush$file1
        mov a,m! ori 1! mov m,a
        lhld pdaddr! mvi c,2! call compare! jnz flush$file1
        lda pdcnt! adi 10h! sta pdcnt! jmp flush$file1

free$files:
        ; free$mode = 1 - remove curdsk files for process
        ;             0 - remove all files for process
        lhld pdaddr! xchg! lxi h,open$root! shld curpos
free$files1:
        lda free$mode! mov b,a
        push d! call searchn$list! pop d! rnz
        lxi b,6! call tst$tbl$lmt! jnz free$files1
        push h! inx h! inx h! inx h
        call test$ffff! jnz free$files2
        call test$ffff! jz free$files3
free$files2:
        mvi a,0ffh! sta incr$pdcnt
free$files3:
        pop h! call remove$locks! call delete$item
        jmp free$files1

remove$locks:
        shld file$id
        inx h! inx h! mov a,m! ani 40h! jz remove$lock3
        push d! lhld prv$pos! push h
        lhld file$id! xchg! lxi h,lock$root! shld cur$pos
remove$lock1:
        mvi b,0! push d! call searchn$list! pop d
        jnz remove$lock2
        lxi b,8! call tst$tbl$lmt! jnz remove$lock1
        call delete$item
        jmp remove$lock1
remove$lock2:
        pop h! shld prv$pos! pop d
remove$lock3:
        lhld file$id! ret

tst$tbl$lmt:
        push h! dad b
        mov a,m! inx h! mov h,m
        sub e! jnz tst$tbl$lmt1
        mov a,h! sub d
tst$tbl$lmt1:
        pop h! ret

create$olist$item:
        mvi b,1! call search$olist
        di
        call create$item! lda attributes! call set$olist$item
        ei
        ret

count$opens:
        xra a! sta open$cnt
        lhld pdaddr! xchg! lxi h,open$root! shld curpos
count$open1:
```

CP/M - PLUS V3.0
COPYRIGHT ©1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
	mvi b,0! push d! call searchn$list! pop d! jnz count$open2
	lxi b,6! call tst$tbl$lmt! jnz count$open1
	lda open$cnt! inr a! sta open$cnt
	jmp count$open1
count$open2:
	lxi h,open$max! lda open$cnt! ret

count$locks:
	xra a! sta lock$cnt
	xchg! lxi h,lock$root! shld cur$pos
count$lock1:
	mvi b,0! push d! call searchn$list! pop d! rnz
	lxi b,8! call tst$tbl$lmt! jnz count$lock1
	lda lock$cnt! inr a! sta lock$cnt
	jmp count$lock1

check$free:
	lda mult$cnt! mov e,a
	mvi d,0! lxi h,free$root! shld cur$pos
check$free1:
	mvi b,0! push d! call searchn$list! pop d! jnz check$free2
	inr d! mov a,d! sub e! jc check$free1
	ret
check$free2:
	pop h! mvi a,14! jmp sta$ret

lock:				; record lock and unlock
	call reselect! call check$fcb
	call test$unlocked
	rz ; file not opened in unlocked mode
	lhld xdmaad! mov e,m! inx h! mov d,m
	xchg! inx h! inx h
	mov a,m! mov b,a! lda curdsk! sub b
	ani 0fh! jnz lock8 ; invalid file id
	mov a,b! ani 40h! jz lock8 ; invalid file id
	dcx h! dcx h! shld file$id
	lda lock$unlock! inr a! jnz lock1 ; jmp if unlock
	call count$locks
	lda lock$cnt! mov b,a
	lda mult$cnt! add b! mov b,a
	lda lock$max! cmp b
	mvi a,12! jc sta$ret ; too many locks by this process
	call check$free
lock1:
	call save$rr! lxi h,lock9! push h! lda mult$cnt
lock2:
	push a! call get$lock$add
	lda lock$unlock! inr a! jnz lock3
	call test$lock
lock3:
	pop a! dcr a! jz lock4
	call incr$rr! jmp lock2
lock4:
	call reset$rr! lda mult$cnt
lock5:
	push a! call get$lock$add
```

```
                lda lock$unlock! inr a! jnz lock6
                call set$lock! jmp lock7
        lock6:
                call free$lock
        lock7:
                pop a! dcr a! rz
                call incr$rr! jmp lock5
        lock8:
                mvi a,13! jmp sta$ret ; invalid file id
        lock9:
                call reset$rr! ret

        get$lock$add:
                lxi h,0! dad sp! shld lock$sp
                mvi a,0ffh! sta lock$shell
                call rseek
                xra a! sta lock$shell
                call getfcb
                lhld aret! mov a,l! ora a! jnz lock$err
                call index! lxi h,1! jz lock$err
                call atran! ret

        lock$perr:
                xra a! sta lock$shell
                xchg! lhld lock$sp! sphl! xchg
        lock$err:
                pop d ; Discard return address
                pop b ; b = mult$cnt-# recs processed
                lda mult$cnt! sub b
                add a! add a! add a! add a
                ora h! mov h,a! mov b,a
                shld aret! ret

        test$lock:
                call move$arecord
                mvi b,3! call search$llist! rnz
                call compare$pds! rz
                lxi h,8! jmp lock$err

        set$lock:
                call move$arecord
                mvi b,1! call search$llist
                di
                call create$item
                xra a! call set$olist$item
                xchg! lhld file$id! xchg
                mov m,d! dcx h! mov m,e
                ei! ret

        free$lock:
                call move$arecord
                mvi b,5! call search$llist! rnz
        free$lock0:
                call delete$item
                mvi b,5! call searchn$list! rnz
                jmp free$lock0
```

```

compare$pds:
        lxi d,6! dad d! xchg
        lxi h,pdaddr! mvi c,2! jmp compare


move$arecord:
        lxi d,arecord! lxi h,packed$dcnt


fix$olist$item:
        lxi d,xdcnt! lxi h,sdcnt
        ; Is xdblk,xdcnt < sdblk,sdcnt
        mvi c,4! ora a!
fix$ol1:
        ldax d! sbb m! inx h! inx d! dcr c! jnz fix$ol1
        rnc
        ; yes - update olist entry
        call swap! call sdcnt$eq$xdcnt
        lxi h,open$root! shld cur$pos
        ; Find file's olist entry
fix$ol2:
        call swap! call pack$sdcnt! call swap
        mvi b,3! call searchn$list! rnz
        ; Update olist entry with new dcnt value
        push h! call pack$sdcnt! pop h
        inx h! inx h! inx h! lxi d,packed$dcnt
        mvi c,3! call move! jmp fix$ol2

hl$eq$hl$and$de:
        mov a,l! ana e! mov l,a
        mov a,h! ana d! mov h,a
        ret

remove$drive:
        xchg! lda curdsk! mov c,a! lxi h,1
        call hlrotl
        mov a,l! cma! ana e! mov e,a
        mov a,h! cma! ana d! mov d,a
        xchg! ret

diskreset:
        lxi h,0! shld ntlog
        xra a! sta set$ro$flag
        lhld info
intrnldiskreset:
        xchg! lhld open$root! mov a,h! ora l! rz
        xchg! lda curdsk! push a! mvi b,0
dskrst1:
        mov a,l! rar! jc dskrst3
dskrst2:
        mvi c,1! call hlrotr! inr b
        mov a,h! ora l! jnz dskrst1
        pop a! sta curdsk
        lhld ntlog! xchg! lhld tlog
        mov a,l! ora e! mov l,a
```

```
                mov a,h! ora d! mov h,a! shld tlog
                inr a! ret
        dskrst3:
                push b! push h! mov a,b! sta curdsk
                lhld rlog! call test$vector1! push a
                lhld rodsk! lda curdsk! call test$vector1! mov b,a
                pop h! lda set$ro$flag! ora b! ora h! sta check$disk
                lxi h,open$root! shld cur$pos
        dskrst4:
                mvi b,1! call searchn$list! jnz dskrst6
                lda check$disk! ora a! jz dskrst5
                push h! call compare$pds! jz dskrst45
                pop h! xra a! xchg! jmp dskrst6
        dskrst45:
                lxi d,ntlog! call set$cdisk
                pop h! jmp dskrst4
        dskrst5:
                lhld info! call remove$drive! shld info
                ori 1
        dskrst6:
                pop h! pop b! jnz dskrst2

                ; error - olist item exists for another process
                ; for removable drive to be reset
                pop a! sta curdsk! mov a,b! adi 41h ; a = ascii drive
                lxi h,6! dad d! mov c,m! inx h! mov b,m ; bc = pdaddr
                push psw! call test$error$mode! pop d! jnz dskrst7
                mov a,d

                push b! push psw
                call rlr! lxi d,console! dad d! mov d,m ; d = console #
                lxi b,deniedmsg! call xprint
                pop psw! mov c,a! call conoutx
                mvi c,':'! call conoutx
                lxi b,cnsmsg! call xprint
                pop h! push h! lxi b,console! dad b
                mov a,m! adi '0'! mov c,a! call conoutx
                lxi b,progmsg! call xprint
                pop h! call dsplynm

        dskrst7:
                pop h ; Remove return addr from diskreset
                lxi h,0ffffh! shld aret ; Flag the error
                ret

        deniedmsg:
                db cr,lf,'disk reset denied, drive ',0
        cnsmsg:
                db ' console ',0
        progmsg:
                db ' program ',0
        endif

        ;
        ;       individual function handlers
        ;
```

```

               func12:
                      ; Return version number

               if MPM
                      lxi h,0100h+dvers! jmp sthl$ret
               else
217D 3AA1FBC372       lda version! jmp sta$ret ; lret = dvers (high = 00)
               endif

               func13:

               if MPM
                      lhld dlog! shld info
                      call diskreset! jz reset$all
                      call reset$37
                      jmp func13$cont
               reset$all:

                      ; Reset disk system - initialize to disk 0
                      lxi h,0! shld rodsk! shld dlog

                      shld rlog! shld tlog
               func13$cont:
                      mvi a,0ffh! sta curdsk
               else
2183 21FFFFCD5A       lxi h,0ffffh! call reset$37x
               endif
2189 AF32DAFB         xra a! sta olddsk ; Note that usrcode remains unchanged

               if MPM
                      xra a! call getmemseg ; a = mem seg tbl index
                      ora a! rz
                      inr a! rz
                      call rlradr! lxi b,msegtbl-rlros! dad b
                      add a! add a! mov e,a! mvi d,0! dad d
                      mov h,m! mvi l,80h
                      jmp intrnlsetdma
               else
218D 21800022D8       lxi h,tbuff! shld dmaad ; dmaad = tbuff
2193 C3B612            jmp setdata ; to data dma address
               endif

               func14:

               if MPM
                      call tmpselect ; seldsk = reg e
                      call rlr! lxi b,diskselect! dad b
                      mov a,m! ani 0fh! rrc! rrc! rrc! rrc
                      mov b,a! lda seldsk! ora b! rrc! rrc! rrc! rrc
                      mov m,a! ret
               else
2196 CD671E           call tmpselect ; seldsk = reg e
2199 3AA42832DA       lda seldsk! sta olddsk! ret
               endif

```

```
                        func15:
                                ; Open file
21A0 CD5411                     call clrmodnum ; Clear the module number

                        if MPM
                                call reselect
                                xra a! sta make$flag
                                call set$sdcnt
                                lxi h,open$file! push h
                                mvi a,0c9h! sta check$fcb4
                                call check$fcb1
                                pop h! lda high$ext! cpi 060h! jnz open$file
                                call home! call set$end$dir
                                jmp open$user$zero
                        open$file:
                                call set$sdcnt
                                call reset$chksum$fcb ; Set invalid check sum
                        else
21A3 CD901E                     call reselectx
                        endif

21A6 CD2D17                     call check$wild ; Check for wild chars in fcb

                        if MPM

                                call get$atts! ani 1100$0000b ; a = attributes
                                cpi 1100$0000b! jnz att$ok
                                ani 0100$0000b ; Mask off unlock mode
                        att$ok:
                                sta high$ext
                                mov b,a! ora a! rar! jnz att$set
                                mvi a,80h
                        att$set:
                                sta attributes! mov a,b
                                ani 80h! jnz call$open
                        endif

21A9 3AE0FBB7CA                 lda usrcode! ora a! jz call$open
21B0 3EFE3218FD                 mvi a,0feh! sta xdcnt+1! inr a! sta search$user0

                        if MPM
                                sta sdcnt0+1
                        endif

                        call$open:
21B9 CDDA17CDE7                 call open! call openx ; returns if unsuccessful, a = 0
21BF 21BD28BEC8                 lxi h,search$user0! cmp m! rz
21C4 773A18FDFE                 mov m,a! lda xdcnt+1! cpi 0feh! rz
                        ;
                        ;       file exists under user 0
                        ;

                        if MPM
                                call swap
                        endif

```

```
21CB CD0A19                     call set$dcnt$dblk

                        if MPM
                                mvi a,0110$0000b
                        else
21CE 3E80                       mvi a,80h
                        endif

21D0 32B528                     sta high$ext
                        open$user$zero:
                                ; Set fcb user # to zero
21D3 2ADBFB3600                 lhld info! mvi m,0
21D8 0E0FCDDD14                 mvi c,namlen! call searchi! call searchn
21E0 CDDD17                     call open1 ; Attempt reopen under user zero
21E3 CDE721                     call openx ; openx returns only if unsuccessful
21E6 C9                         ret
                        openx:
21E7 CDF812C8                   call end$of$dir! rz
21EB CD4F107E3C                 call getfcba! mov a,m! inr a! jnz openxa
21F3 1B1B1A77                   dcx d! dcx d! ldax d! mov m,a
                        openxa:
                                ; open successful
21F7 E1                         pop h ; Discard return address
                                ; Was file opened under user 0 after unsuccessful
                                ; attempt to open under user n?

                        if MPM
                                lda high$ext! cpi 060h! jz openx00 ; yes
                                ; Was file opened in locked mode?
                                ora a! jnz openx0 ; no
                                ; does user = zero?
                                lhld info! ora m! jnz openx0 ; no
                                ; Does file have read/only attribute set?
                                call rotest! jnc openx0 ; no
                                ; Does file have system attribute set?
                                inx h! mov a,m! ral! jnc openx0 ; no

                                ; Force open mode to read/only mode and set user 0 flag
                                ; if file opened in locked mode, user = 0, and
                                ; file has read/only and system attributes set

                        openx00:

                        else
21F8 3AB52817D2                 lda high$ext! ral! jnc openx0
                        endif

                                ; Is file under user 0 a system file ?

                        if MPM
                                mvi a,20h! sta attributes
                        endif

21FF 2ADBFB110A                 lhld info! lxi d,10! dad d
2206 7EE680C212                 mov a,m! ani 80h! jnz openx0 ; yes - open successful
                                ; open fails
```

```
220C 32B528C3C0         sta high$ext! jmp lret$eq$ff
                openx0:

                if MPM
                        call reset$chksum$fcb
                else
2212 CD4112             call set$lsn
                endif

                if BANKED

                        ; Are passwords enabled on drive?
2215 CD731FE680         call get$dir$mode! ani 80h! jz openx1a ; no
                        ; Is this 1st dir fcb?
221D CD4021C24E         call qdirfcb1! jnz openx0a ; no
                        ; Does sfcb exist?
2223 CD1021B7C2         call get$dtba$8! ora a! jnz openx0a ; no
                        ; Is sfcb password mode read or write?
222A 7EE6C0CA76         mov a,m! ani 0c0h! jz openx1a ; no
                        ; Does xfcb exist?
2230 CD2416             call xdcnt$eq$dcnt
2233 CD8320C257         call get$xfcb! jnz openx0b ; yes
                        ; no - set sfcb password mode to zero
2239 CD2B16C8           call restore$dir$fcb! rz ; (error)
                        ; Does sfcb still exist?
223D CD1021B7C2         call get$dtba$8! ora a! jnz openx1a ; no (error)
                        ; sfcb password mode = 0
2244 77                 mov m,a
                        ; update sfcb
2245 CD1A11CC9A         call nowrite! cz seek$copy
224B C37622             jmp openx1a
                openx0a:
224E CD2416             call xdcnt$eq$dcnt
                        ; Does xfcb exist?
2251 CD8320CA72         call get$xfcb! jz openx1 ; no
                openx0b:
                        ; yes - check password
2257 CD0220CA72         call cmp$pw! jz openx1
225D CD851F             call chk$pw$error
2260 3AE428E6C0         lda pw$mode! ani 0c0h! jz openx1
2268 E680C2DF1F         ani 80h! jnz pw$error
226D 3E8032B628         mvi a,080h! sta xfcb$read$only
                openx1:
2272 CD2B16C8           call restore$dir$fcb! rz ; (error)
                openx1a:
2276 CD4112             call set$lsn

                if MPM
                        call pack$sdcnt
                        ; Is this file currently open?
                        mvi b,3! call search$olist! jz openx04
                openx01:
                        ; no - is olist full?
                        lhld free$root! mov a,l! ora h! jnz openx03
                        ; yes - error
                openx02:
                      mvi a,11! jmp set$aret
               openx03:
                      ; Has process exceeded open file maximum?
                      call count$opens! sub m! jc openx035
                      ; yes - error
               openx034:
                      mvi a,10! jmp set$aret
               openx035:
                      ; Create new olist element
                      call create$olist$item
                      jmp openx08
               openx04:
                      ; Do file attributes match?
                      inx h! inx h
                      lda attributes! ora m! cmp m! jnz openx06
                      ; yes - is open mode locked?
                      ani 80h! jnz openx07
                      ; no - has this file been opened by this process?
                      lhld prv$pos! shld cur$pos
                      mvi b,5! call searchn$list! jnz openx01
               openx05:
                      ; yes - increment open file count
                      lxi d,8! dad d! inr m! jnz openx08
                      ; count overflow
                      inx h! inr m! jmp openx08
               openx06:
                      ; error - file opened by another process in incompatible mode
                      mvi a,5! jmp set$aret
               openx07:
                      ; Does this olist item belong to this process?
                      dcx h! dcx h! push h
                      call compare$pds
                      pop h! jnz openx06 ; no - error
                      jmp openx05 ; yes
               openx08:; Wopen ok
                      ; Was file opened in unlocked mode?
                      lda attributes! ani 40h! jz openx09 ; no
                      ; yes - return .olist$item in ranrec field of fcb
                      call get$rra
                      lxi d,cur$pos! mvi c,2! call move
               openx09:
                      call set$fcb$cks$flag
                      lda make$flag! ora a! rnz
               endif
               endif

2279 0E40             mvi c,0100$0000b
               openx2:
227B CD3221CCDD       call qstamp! cz stamp1
2281 1190FBC308       lxi d,olog! jmp set$cdisk

               func16:
                      ; Close file
2287 CD9A1E           call reselect

               if MPM
```

```
                                call get$atts! sta attributes
                                lxi h,close00! push h
                                mvi a,0c9h! sta check$fcb4
                                call check$fcb1! pop h
                                call set$sdcnt
                                call getmodnum! ani 80h! jnz close01
                                call close! jmp close02
                      close00:
                                mvi a,6! jmp set$aret
                      close01:
                                mvi a,0ffh! sta dont$close! call close1
                      close02:
                      else
228A CD4112                     call set$lsn
228D CDD810CD3D                 call chek$fcb! call close
                      endif

2293 3A9F0D3CC8                 lda lret! inr a! rz

2298 C31A25                     jmp flush ; Flush buffers

                      if MPM
                                lda attributes! ral! rc
                                call pack$sdcnt
                                ; Find olist item for this process & file
                                mvi b,5! call search$olist! jnz close03
                                ; Decrement open count
                                push h! lxi d,8! dad d
                                mov a,m! sui 1! mov m,a! inx h
                                mov a,m! sbi 0! mov m,a! dcx h
                                ; Is open count = 0ffffh
                                call test$ffff! pop h! jnz close03
                                ; yes - remove file's olist entry
                                shld file$id! call delete$item
                                call reset$chksum$fcb
                                ; if unlocked file, remove file's locktbl entries
                                call test$unlocked! jz close03
                                lhld file$id! call remove$locks
                      close03:
                                ret

                      endif

                      func17:
                                ; Search for first occurrence of a file
229B EBAF                       xchg! xra a
                      csearch:
229D F5                         push a
229E 7EFE3FC2AF                 mov a,m! cpi '?'! jnz csearch1 ; no reselect if ?
22A4 CD6B1ECDE9                 call curselect! call noselect0! mvi c,0! jmp csearch3
                      csearch1:
22AF CD3F107EFE                 call getexta! mov a,m! cpi '?'! jz csearch2
22B8 CD5A11CD54                 call clr$ext! call clrmodnum
                      csearch2:
22BE CD901E                     call reselectx
22C1 0E0F                       mvi c,namlen
```

```
                    csearch3:
22C3 F1F5CADA22             pop a! push a! jz csearch4
                            ; dcnt = dcnt & 0fch
22C8 2AE1FBE53E             lhld dcnt! push h! mvi a,0fch
22CE A56F22E1FB             ana l! mov l,a! shld dcnt
22D3 CD9212                 call rd$dir
22D6 E122E1FB               pop h! shld dcnt
                    csearch4:
22DA F1                     pop a
22DB 21C312                 lxi h,dir$to$user
22DE E5                     push h
22DF CAF714                 jz search
22E2 3AE5FB4FCD             lda searchl! mov c,a! call searchi! jmp searchn

                    func18:
                            ; Search for next occurrence of a file name

                    if BANKED
22EC EB22A028               xchg! shld searcha
                    else
                            lhld searcha! shld info
                    endif

22F0 F601C39D22             ori 1! jmp csearch

                    func19:
                            ; Delete a file
22F5 CD901E                 call reselectx
22F8 C33616                 jmp delete

                    func20:
                            ; Read a file
22FB CD9A1E                 call reselect
22FE CDCE10                 call check$fcb
2301 C3F519                 jmp seqdiskread

                    func21:
                            ; Write a file
2304 CD9A1E                 call reselect
2307 CDCE10                 call check$fcb
230A C3631B                 jmp seqdiskwrite

                    func22:
                            ; Make a file

                    if BANKED
230D CD181032B3             call get$atts! sta attributes
                    endif

2313 CD5A11                 call clr$ext
2316 CD5411                 call clrmodnum ; fcb mod = 0
2319 CD901E                 call reselectx

                    if MPM
                            call reset$chksum$fcb
                    endif
```

```

231C CD2D17            call check$wild
231F CD0319            call set$xdcnt ; Reset xdcnt for make

                if MPM
                       call set$sdcnt
                endif

2322 CDDA17            call open ; Verify file does not already exist

                if MPM
                       call reset$chksum$fcb
                endif

                       ; Does dir fcb for fcb exist?
                       ; ora a required to reset carry
2325 CDF812B7CA        call end$of$dir! ora a! jz makea0 ; no
                       ; Is dir$ext < fcb(ext)?
232C CD9714BED2        call get$dir$ext! cmp m! jnc file$exists ; no
                makea0:
2333 F5                push a ; carry set if dir fcb already exists

                if MPM
                       lda attributes! ani 80h! rrc! jnz makex00
                       mvi a,80h
                makex00:
                       sta make$flag
                       lda sdcnt+1! inr a! jz makex01
                       call pack$sdcnt
                       mvi b,3! call search$olist! jz make$x02
                makex01:
                       lhld free$root! mov a,l! ora h! jz openx02
                       jmp makex03
                makex02:
                       inx h! inx h
                       lda makeflag! ana m! jz openx06
                       dcx h! dcx h! call compare$pds! jz makex03
                       lda makeflag! ral! jc openx06
                makex03:

                endif

                if BANKED
                       ; Is fcb 1st fcb for file?
2334 CD4021CA59        call qdirfcb1! jz makex04 ; yes
                       ; no - does dir lbl require passwords?
233A CD731FE680        call get$dir$mode! ani 80h! jz makex04
                       ; no - does xfcb exist with mode 1 or 2 password?
2342 CD8320CA59        call get$xfcb! jz makex04
                       ; yes - check password
2348 CDD720CA59        call chk$xfcb$password1! jz makex04
                       ; Verify password error
234E CD851F            call chk$pw$error
2351 3AE428E6C0        lda pw$mode! ani 0c0h! jnz pw$error
                makex04:

```

```
                endif

                        ; carry on stack indicates a make not required because
                        ; of extent folding
2359 F1D41619           pop a! cnc make

                if MPM
                        call reset$chksum$fcb
                endif

                        ; end$of$dir call either applies to above make or open call
235D CDF812C8           call end$of$dir! rz ; Return if make unsuccessful

                if not MPM
2361 CD4112             call set$lsn
                endif

                if BANKED

                        ; Are passwords activated by dir lbl?
2364 CD731FE680         call get$dir$mode! ani 80h! jz make3a
                        ; Did user set password attribute?
236C 3AB328E640         lda attributes! ani 40h! jz make3a
                        ; Is fcb file's 1st logical fcb?
2374 CD4021C2D2         call qdirfcb1! jnz make3a
                        ; yes - does xfcb already exist for file
237A CD2416             call xdcnt$eq$dcnt
237D CD8320C297         call get$xfcb! jnz make00 ; yes
                        ; Attempt to make xfcb
2383 3EFF3215FD         mvi a,0ffh! sta make$xfcb! call make! jnz make00
                        ; xfcb make failed - delete fcb that was created above
238E CDF014             call search$namlen
2391 CD8716C3C0         call delete10! jmp lret$eq$ff ; Return with a = 0ffh

                make00:
2397 CDB120             call init$xfcb ; Initialize xfcb
                        ; Get password mode from dma + 8
239A EB2A19FD01         xchg! lhld xdmaad! lxi b,8! dad b! xchg
23A3 1AE6E0C2AB         ldax d! ani 0e0h! jnz make2
23A9 3E80               mvi a,080h ; default password mode is read protect
                make2:
23AB 32E428             sta pw$mode
                        ; Set xfcb password mode field
23AE F5CD9920F1         push a! call getxfcb1! pop a! mov m,a
                        ; Set xfcb password and password checksum
                        ; Fix hash table and write xfcb
23B4 CD422070CD         call set$pw! mov m,b! call sdl3
                        ; Return to fcb
23BB CD2B16C8           call restore$dir$fcb! rz
                        ; Does sfcb exist?
23BF 0E08CD1221         mvi c,8! call getdtba! ora a! jnz make3a ; no
                        ; Place password mode in sfcb if sfcb exists
23C8 3AE42877CD         lda pw$mode! mov m,a! call seek$copy
23CF CD4112             call set$lsn
                endif

```

```
                        make3a:
23D2 0E50                       mvi c,0101$0000b

                        if MPM
                                call openx2
                                lda make$flag! sta attributes
                                ani 40h! ral! sta high$ext
                                lda sdcnt+1! inr a! jnz makexx02
                                call sdcnt$eq$xdcnt! call pack$sdcnt
                                jmp openx03
                        makexx02:
                                call fix$olist$item! jmp openx1
                                jmp set$fcb$cks$flag
                        else
23D4 CD7B22                     call openx2
23D7 0E20CD3221                 mvi c,0010$0000b! call qstamp! rnz
23DD CDE220C357                 call stamp2! jmp set$filewf
                        endif

                        file$exists:
23E3 3E08                       mvi a,8
                        set$aret:
23E5 4F32A00DCD                 mov c,a! sta aret+1! call lret$eq$ff

                        if MPM
                                call test$error$mode! jnz goback
                        else
23EC C3B80D                     jmp goerr1
                        endif

                        if MPM
                                mov a,c! sui 3
                                mov l,a! mvi h,0! dad h
                                lxi d,xerr$list! dad d
                                mov e,m! inx h! mov d,m
                                xchg! jmp report$err
                        endif

                        func23:
                                ; Rename a file
23EF CD901E                     call reselectx
23F2 C35217                     jmp rename

                        func24:
                                ; Return the login vector
23F5 2A6528C329                 lhld dlog! jmp sthl$ret

                        func25:
                                ; Return selected disk number
23FB 3AA428C372                 lda seldsk! jmp sta$ret

                        func26:

                        if MPM
                                ; Save dma address in process descriptor
                                lhld info
```

```
                intrnlsetdma:
                        xchg
                        call rlr! lxi b,disksetdma! dad b
                        mov m,e! inx h! mov m,d
                endif

                        ; Set the subsequent dma address to info
2401 EB22D8FB           xchg! shld dmaad ; dmaad = info
2405 C3B612              jmp setdata ; to data dma address

                func27:
                        ; Return the login vector address
2408 CD6B1E             call curselect
240B 2A7728C329         lhld alloca! jmp sthl$ret

                if MPM

                func28:
                        ; Write protect current disk
                        ; first check for open files on disk
                        mvi a,0ffh! sta set$ro$flag
                        lda seldsk! mov c,a! lxi h,0001h
                        call hlrotl! call intrnldiskreset
                        jmp set$ro
                else

11D3 =          func28:         equ     set$ro ; Write protect current disk

                endif

                func29:
                        ; Return r/o bit vector
2411 2A6328C329         lhld rodsk! jmp sthl$ret

                func30:
                        ; Set file indicators
2417 CD2D17             call check$wild
241A CD901E             call reselectx
241D CDA417             call indicators
2420 C38014             jmp copy$dirloc ; lret=dirloc

                func31:
                        ; Return address of disk parameter block
2423 CD6B1E             call curselect
2426 2A7328             lhld dpbaddr
                sthl$ret:
2429 229F0DC9           shld aret! ret

                func32:
                        ; Set user code
242D 3A9E28FEFF          lda linfo! cpi 0ffh! jnz setusrcode
                                ; Interrogate user code instead
2435 3AE0FBC372                 lda usrcode! jmp sta$ret ; lret=usrcode
                        setusrcode:
243B E60F32E0FB                 ani 0fh! sta usrcode

```

```
                if MPM
                                push a
                                call rlr! lxi b,diskselect! dad b
                                pop b
                                mov a,m! ani 0f0h! ora b! mov m,a
                endif

2440 C9                         ret

                func33:
                        ; Random disk read operation
2441 CD9A1E             call reselect
2444 CDCE10             call check$fcb
2447 C3B91D             jmp randiskread ; to perform the disk read

                func34:
                        ; Random disk write operation
244A CD9A1E             call reselect
244D CDCE10             call check$fcb
2450 C3C21D             jmp randiskwrite ; to perform the disk write

                func35:
                        ; Return file size (0-262,144)
2453 CD9A1E             call reselect
2456 C3131E             jmp getfilesize

1E40 =          func36 equ setrandom ; Set random record

                func37:
                        ; Drive reset

                if MPM
                        call diskreset
                reset$37:
                        lhld info
                else
2459 EB                 xchg
                endif

                reset$37x:
245A 7D2F5F7C2F         mov a,l! cma! mov e,a! mov a,h! cma
245F 2A6528A457         lhld dlog! ana h! mov d,a! mov a,l! ana e
2466 5F2A6328EB         mov e,a! lhld rodsk! xchg! shld dlog

                if MPM
                        push h! call hl$eq$hl$and$de
                else
246E 7DA36F             mov a,l! ana e! mov l,a
2471 7CA267             mov a,h! ana d! mov h,a
                endif

2474 226328             shld rodsk

                if MPM
                        pop h! xchg! lhld rlog! call hl$eq$hl$and$de! shld rlog
                endif
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```

                       ; Force select call in next curselect
2477 3EFF32A928        mvi a,0ffh! sta curdsk! ret

                if MPM

                func38:
                       ; Access drive

                       lxi h,packed$dcnt! mvi a,0ffh
                       mov m,a! inx h! mov m,a! inx h! mov m,a
                       xra a! xchg! lxi b,16
                acc$drv0:
                       dad h! adc b! dcr c! jnz acc$drv0
                       ora a! rz
                       sta mult$cnt! dcr a! push a
                       call acc$drv02
                       pop a! jmp openx02 ; insufficient free lock list items
                acc$drv02:
                       call check$free! pop h ; Discard return addr, free space exists
                       call count$opens! pop b! add b! jc openx034
                       sub m! jnc openx034 ; openmax exceeded
                       lhld info! lda curdsk! push a! mvi a,16
                acc$drv1:
                       dcr a! dad h! jc acc$drv2
                acc$drv15:
                       ora a! jnz acc$drv1
                       pop a! sta curdsk! ret
                acc$drv2:
                       push a! push h! sta curdsk
                       call create$olist$item
                       pop h! pop a! jmp acc$drv15

                func39:
                       ; Free drive
                       lhld open$root! mov a,h! ora l! rz
                       xra a! sta incr$pdcnt! inr a! sta free$mode
                       lhld info! mov a,h! cmp l! jnz free$drv1
                       inr a! jnz free$drv1
                       sta free$mode! call free$files! jmp free$drv3
                free$drv1:
                       lda curdsk! push a! mvi a,16
                free$drv2:
                       dcr a! dad h! jc free$drv4
                free$drv25:
                       ora a! jnz free$drv2
                       pop a! sta curdsk
                free$drv3:
                       lda incr$pdcnt! ora a! rz
                       lda pdcnt! jmp chk$olist1
                free$drv4:
                       push a! push h! sta curdsk
                       call free$files
                       pop h! pop a! jmp free$drv25
                else

```

```
0A75 =          func38          equ     func$ret
0A75 =          func39 equ      func$ret

                endif

244A =          func40  equ     func34          ; Write random with zero fill

                if MPM

                func41 equ      func$ret        ; Test & write
                func42:                                 ; Record lock
                        mvi a,0ffh! sta lock$unlock! jmp lock
                func43:                                 ; Record unlock
                        xra a! sta lock$unlock! jmp lock

                else

0A75 =          func42 equ      func$ret        ; Record lock
0A75 =          func43 equ      func$ret        ; Record unlock

                endif

                func44:                                 ; Set multi-sector count
247D 7BB7CAC015         mov a,e! ora a! jz lret$eq$ff
2482 FE81D2C015         cpi 129! jnc lret$eq$ff
2487 32E6FB             sta mult$cnt

                if MPM
                        mov d,a
                        call rlr! lxi b,mult$cnt$off! dad b
                        mov m,d
                endif

248A C9                 ret

                func45:                                 ; Set bdos error mode

                if MPM
                        call rlr! lxi b,pname+4! dad b
                        call set$pflag
                        mov m,a! inx h
                        call set$pflag
                        mov m,a! ret

                set$pflag:
                        mov a,m! ani 7fh! inr e! rnz
                        ori 80h! ret
                else
248B 7B32E7FB           mov a,e! sta error$mode
                endif

248F C9                 ret

                func46:
                        ; Get free space
                        ; Perform temporary select of specified drive
```

```
2490 CD671E             call tmpselect
2493 2A7728EB           lhld alloca! xchg ; de = alloc vector addr
2497 CDDE13             call get$nalbs ; Get # alloc blocks
                        ; hl = # of allocation vector bytes
                        ; Count # of true bits in allocation vector
249A 010000             lxi b,0 ; bc = true bit accumulator
249D 1A         gsp1:   ldax d
249E B7CAAA24   gsp2:   ora a! jz gsp4
24A2 1FD2A224   gsp3:   rar! jnc gsp3
24A6 03C39E24           inx b! jmp gsp2
24AA 132B       gsp4:   inx d! dcx h
24AC 7DB4C29D24         mov a,l! ora h! jnz gsp1
                        ; hl = 0 when allocation vector processed
                        ; Compute maxall + 1 - bc
24B1 2A852823           lhld maxall! inx h
24B5 7D916F             mov a,l! sub c! mov l,a
24B8 7C9867             mov a,h! sbb b! mov h,a
                        ; hl = # of available blocks on drive
24BB 3A82284FAF         lda blkshf! mov c,a! xra a
24C0 CDF60D             call shl3bv
                        ; ahl = # of available sectors on drive
                        ; Store ahl in beginning of current dma
24C3 EB2A19FD73         xchg! lhld xdmaad! mov m,e! inx h
24C9 722377C9           mov m,d! inx h! mov m,a! ret

                if MPM

                func47 equ      func$ret

                else

                func47:                         ; Chain to program
24CD 21B3FB7EF6         lxi h,ccp$flgs! mov a,m! ori 80h! mov m,a
24D4 1CC22F03           inr e! jnz rebootx1
24D8 7EF64077           mov a,m! ori 40h! mov m,a
24DC C32F03             jmp rebootx1
                endif

                func48:                                 ; Flush buffers
24DF CD2F1F             call check$all$media
24E2 CD48FF             call flushf
24E5 CDA50E             call diocomp
                flush0:                                 ; Function 98 entry point
24E8 2A65283E10         lhld dlog! mvi a,16
                flush1:
24ED 3D29D21525         dcr a! dad h! jnc flush5
24F2 F5E55FCD67         push a! push h! mov e,a! call tmpselect ; seldsk = e
24F8 3ADFFBFE30         lda fx! cpi 48! jz flush3
                        ; Function 98 - reset allocation
                        ; Copy 2nd ALV over 1st ALV
2500 CD7713C310         call copy$alv! jmp flush35
                flush3:
2506 CD2025             call flushx
                        ; if e = 0ffh then discard buffers after possible flush
2509 3A9E283CC2         lda linfo! inr a! jnz flush4
                flush35:
```

```
2510 CD5212           call discard$data
                flush4:
2513 E1F1             pop h! pop a
                flush5:
2515 B7C2ED24         ora a! jnz flush1
2519 C9               ret

                flush:
251A CD48FF           call flushf
251D CDA50E           call diocomp
                flushx:
2520 3A9028B7C8       lda phymsk! ora a! rz
2525 3E04C3FF28       mvi a,4! jmp deblock$dta

                if MPM

                func49 equ     func$ret

                else

                func49:        ; Get/Set system control block

252A EB7EFE63D0       xchg! mov a,m! cpi 99! rnc
252F EB219CFB85       xchg! lxi h,scb! add l! mov l,a
2535 EB237EFEFE       xchg! inx h! mov a,m! cpi 0feh! jnc func49$set
253D EB5E2356EB       xchg! mov e,m! inx h! mov d,m! xchg
2542 C32924           jmp sthl$ret
                func49$set:
2545 47237E1204       mov b,a! inx h! mov a,m! stax d! inr b! rz
254B 23137E12C9       inx h! inx d! mov a,m! stax d! ret
                endif

                if MPM

                func50 equ     func$ret

                else

                func50:                             ; Direct bios call
                      ; de -> function (1 byte)
                      ;       a  value (1 byte)
                      ;       bc value (2 bytes)
                      ;       de value (2 bytes)
                      ;       hl value (2 bytes)

2550 21B025E5         lxi h,func50$ret! push h
2554 EB               xchg

                if BANKED
2555 7EFE1BC8         mov a,m! cpi 27! rz
2559 FE0CC26225       cpi 12! jnz dir$bios1
255E 11B825D5         lxi d,dir$bios3! push d
                dir$bios1:
2562 FE09C26B25       cpi 9! jnz dir$bios2
2567 119025D5         lxi d,dirbios4! push d
                dir$bios2:
```

```

                endif

256B E52323            push h! inx h! inx h
256E 4E234623          mov c,m! inx h! mov b,m! inx h
2572 5E235623          mov e,m! inx h! mov d,m! inx h
2576 7E23666F          mov a,m! inx h! mov h,m! mov l,a
257A E37EE56F87        xthl! mov a,m! push h! mov l,a! add a! add l

2580 2100FF            lxi h,bios

2583 856FE3            add l! mov l,a! xthl
2586 237EE1E3C9        inx h! mov a,m! pop h! xthl! ret

                if BANKED

                dir$bios3:
258B 3E01C354FF        mvi a,1! jmp setbnkf

                dir$bios4:
2590 7DB4C8            mov a,l! ora h! rz
2593 EB210A0019        xchg! lxi h,10! dad d! mvi m,0 ; Zero login sequence #
259A 2AF9FBCD7F        lhld common$base! call subdh! xchg! rnc
                       ; Copy DPH to common memory
25A2 EB2ADBFB23        xchg! lhld info! inx h! push h! lxi b,25
25AB CD4BFFE1C9        call movef! pop h! ret
                endif

                func50$ret:

                if BANKED
25B0 229F0D47          shld aret! mov b,a
25B4 2ADBFB7E          lhld info! mov a,m
25B8 FE09C8            cpi 9! rz
25BB FE10C8            cpi 16! rz
25BE FE14C8            cpi 20! rz
25C1 FE16C8            cpi 22! rz
25C4 78C3720A          mov a,b! jmp sta$ret
                else
                       xchg! lhld entsp! sphl! xchg! ret
                endif
                endif

24E8 =          func98        equ    flush0              ; Reset Allocation

                func99:                                  ; Truncate file
25C8 CD901E            call reselectx
25CB CD2D17            call check$wild

                if BANKED
25CE CD781FC485        call chk$password! cnz chk$pw$error
                endif

25D4 0EFFCDE41C        mvi c,true! call rseek! jnz lret$eq$ff
                       ; compute dir$fcb size
25DC CD4011110F        call getdptra! lxi d,reccnt
```

```
25E2 CDCB1D     call compute$rr ; cba = fcb size
                ; Is random rec # >= dir$fcb size
25E5 CD3710CDFE call get$rra! call compare$rr
25EB DAC015     jc lret$eq$ff ; yes ( > )
25EE B2CAC015   ora d! jz lret$eq$ff  ; yes ( = )
                ; Perform truncate
25F2 CD2811     call check$rodir ; may be r/o file
25F5 CD9A12     call wrdir ; verify BIOS can write to disk
25F8 CD5321     call update$stamp ; Set update stamp
25FB CDF514     call search$extnum
           trunc1:
25FE CA8014     jz copy$dirloc
                ; is dirfcb < fcb?
2601 CD7126DA1C call compare$mod$ext! jc trunc2 ; yes
                ; remove dirfcb blocks from allocation vector
2607 F50E00CD8B push a! mvi c,0! call scandm$ab! pop a
                ; is dirfcb = fcb?
260E CA2226     jz trunc3 ; yes
                ; delete dirfcb
2611 CD401136E5 call getdptra! mvi m,empty! call fix$hash
           trunc15:
2619 CD9A12     call wrdir
           trunc2:
261C CD0315     call searchn
261F C3FE25     jmp trunc1
           trunc3:
2622 CD5810CDA4 call getfcb! call dm$position
2628 CD8B26     call zero$dm
                ; fcb(extnum) = dir$ext after blocks removed
262B CD9714BE77 call get$dir$ext! cmp m! mov m,a! push a
                ; fcb(rc) = fcb(cr) + 1
2631 CD4F107E3C call getfcba! mov a,m! inr a! stax d
                ; rc = 0 or 128 if dir$ext < fcb(extnum)
2637 F1EBC41518 pop a! xchg! cnz set$rc3
                ; rc = 0 if no blocks remain in fcb
263C 3A9F2BB7CC lda dminx! ora a! cz set$rc3
2643 010B00CD67 lxi b,11! call get$fcb$adds! xchg
                ; reset archive (t3') attribute bit
264A 7EE67F7723 mov a,m! ani 7fh! mov m,a! inx h! inx d
                ; dirfcb(extnum) = fcb(extnum)
2650 1A77       ldax d! mov m,a
                ; advance to .fcb(reccnt) & .dirfcb(reccnt)
2652 2336002323 inx h! mvi m,0! inx h! inx h
2657 131313     inx d! inx d! inx d
                ; dirfcb_rc+dskmap = fcb_rc+dskmap
265A 0E11CD290E mvi c,17! call move
                ; restore non-erased blkidxs in allocation vector
265F 0E01CD8B13 mvi c,1! call scandm$ab
2664 C31926     jmp trunc15

           get$fcb$adds:
2667 CD401109EB call getdptra! dad b! xchg
266C 2ADBFB09C9 lhld info! dad b! ret

           compare$mod$ext:
2671 010E00CD67 lxi b,modnum! call get$fcb$adds
```

```
2677 7EE63F47              mov a,m! ani 3fh! mov b,a
                           ; compare dirfcb(modnum) to fcb(modnum)
267B 1AB8C0                ldax d! cmp b! rnz ; dirfcb(modnum) ^= fcb(modnum)
267E 2B2B1B1B              dcx h! dcx h! dcx d! dcx d
                           ; compare dirfcb(extnum) to fcb(extnum)
2682 1A4ECD8614            ldax d! mov c,m! call compext! rz ; dirfcb(extnum) = fcb(extnum)
2688 1ABEC9                ldax d! cmp m! ret

                   zero$dm:
268B 3C21A32834            inr a! lxi h,single! inr m! jz zero$dm1
2693 87                    add a
                   zero$dm1:
2694 35                    dcr m
2695 CDC40F4F06            call getdma! mov c,a! mvi b,0! dad b
269C 3E10                  mvi a,16
                   zero$dm2:
269E B9C8                  cmp c! rz
26A0 70230CC39E            mov m,b! inx h! inr c! jmp zero$dm2

                   if BANKED

                   func100:                      ; Set directory label
                           ; de -> .fcb
                           ;       drive location
                           ;       name & type fields user's discretion
                           ;       extent field definition
                           ;       bit 1 (80h): enable passwords on drive
                           ;       bit 2 (40h): enable file access
                           ;       bit 3 (20h): enable file update stamping
                           ;       bit 4 (10h): enable file create stamping
                           ;       bit 8 (01h): assign new password to dir lbl
26A6 CD901E                call reselectx
26A9 2ADBFB3621            lhld info! mvi m,21h! mvi c,1
26B0 CDF714C2BF            call search! jnz sdl0
26B6 CD3F107EE6            call getexta! mov a,m! ani 0111$0000b! jnz lret$eq$ff
                   sdl0:
                           ; Does dir lbl exist on drive?
26BF 2ADBFB3620            lhld info! mvi m,20h! mvi c,1
26C6 CD0319CDF7            call set$xdcnt! call search! jnz sdl1
                           ; no - make one
26CF 3EFF3215FD            mvi a,0ffh! sta make$xfcb
26D4 CD1619C8              call make! rz ; no dir space
26D8 CDB120                call init$xfcb
26DB 011800CD05            lxi b,24! call stamp5! call stamp1
                   sdl1:
                           ; Update date & time stamp
26E4 011C00CD05            lxi b,28! call stamp5! call stamp2
                           ; Verify password - new dir lbl falls through
26ED CDD420C2DF            call chk$xfcb$password! jnz pw$error
26F3 010000CDB7            lxi b,0! call init$xfcb0
                           ; Set dir lbl dta in extent field
26F9 1AF60177              ldax d! ori 1h! mov m,a
                           ; Low bit of dir lbl data set to indicate dir lbl exists
                           ; Update drive's dir lbl vector element
26FD E52A6F2877            push h! lhld drvlbla! mov m,a! pop h
                   sdl2:
                      ; Assign new password to dir lbl or xfcb?
2703 1AE601CA19       ldax d! ani 1! jz sdl3
                      ; yes - new password field is in 2nd 8 bytes of dma
2709 110800CDA7       lxi d,8! call adjust$dmaad
270F CD422070         call set$pw! mov m,b
2713 11F8FFCDA7       lxi d,-8! call adjust$dmaad
               sdl3:
2719 CDC82C           call fix$hash
271C C39A12           jmp seek$copy
               else

               func100        equ     lret$eq$ff
               func103 equ    lret$eq$ff

               endif

               func101:
                      ; Return directory label data
                      ; Perform temporary select of specified drive
271F CD671E           call tmpselect
2722 CD731FC372       call get$dir$mode! jmp sta$ret

               func102:
                      ; Read file xfcb
2728 CD901E           call reselectx
272B CD2D17           call check$wild
272E CD9C10           call zero$ext$mod
2731 CDF014C8         call search$namlen! rz
2735 CDC40F0108       call getdma! lxi b,8! call zero
273E E50E00CD12       push h! mvi c,0! call get$dtba! ora a! jnz rxfcb2
2748 D1EB0E08         pop d! xchg! mvi c,8

               if BANKED
274C CD290E1AC3       call move! ldax d! jmp rxfcb3
               else
                      jmp move
               endif

               rxfcb2:
2753 E1010800         pop h! lxi b,8

               if BANKED
2757 CDA410CD83       call zero! call get$xfcb! rz
275E 7E               mov a,m
               rxfcb3:
275F CD3F1077C9       call getexta! mov m,a! ret
               else
                      jmp zero
               endif

               if BANKED

               func103:
                      ; Write or update file xfcb
2764 CD901E           call reselectx
                      ; Are passwords enabled in directory label?
```

```
2767 CD731F17D2        call get$dir$mode! ral! jnc lret$eq$ff ; no
276E CD2D17            call check$wild
                       ; Save .fcb(ext) & ext
2771 CD3F1046E5        call getexta! mov b,m! push h! push b
                       ; Set extent & mod to zero
2777 CD9C10            call zero$ext$mod
                       ; Does file's 1st fcb exist in directory?
277A CDF014            call search$namlen
                       ; Restore extent
277D C1E170C8          pop b! pop h! mov m,b! rz ; no
2781 CD0319            call set$xdcnt
                       ; Does sfcb exist?
2784 CD1021B7CA        call get$dtba$8! ora a! jz wxfcb5 ; yes
                       ; No - Does xfcb exist?
278B CD8320C2A1        call get$xfcb! jnz wxfcb1 ; yes
                wxfcb0:
                       ; no - does file exist in directory?
2791 3EFF3215FD        mvi a,0ffh! sta make$xfcb
2796 CDF514C8          call search$extnum! rz
                       ; yes - attempt to make xfcb for file
279A CD1619C8          call make! rz ; no dir space
                       ; Initialize xfcb
279E CDB120            call init$xfcb
                wxfcb1:
                       ; Verify password - new xfcb falls through
27A1 CDD420C2DF        call chk$xfcb$password! jnz pw$error
                       ; Set xfcb options data
27A7 E5CD3F10D1        push h! call getexta! pop d! xchg
27AD 7EB7C2BE27        mov a,m! ora a! jnz wxfcb2
27B2 1AE601C2BE        ldax d! ani 1! jnz wxfcb2
27B8 CD1927C3CA        call sdl3! jmp wxfcb4
                wxfcb2:
27BE 1AE6E0C2C6        ldax d! ani 0e0h! jnz wxfcb3
27C4 3E80              mvi a,80h
                wxfcb3:
27C6 77CD0327          mov m,a! call sdl2
                wxfcb4:
27CA CD99203D32        call get$xfcb1! dcr a! sta pw$mode
27D1 CD9C10            call zero$ext$mod
27D4 CDF014C8          call search$namlen! rz
27D8 CD1021B7C0        call get$dtba$8! ora a! rnz
27DD 3AE42877C3        lda pw$mode! mov m,a! jmp seek$copy
                wxfcb5:
                       ; Take sfcb's password mode over xfcb's mode
27E4 7EF5              mov a,m! push a
27E6 CD8320            call get$xfcb
                       ; does xfcb exist?
27E9 C1CA9127          pop b! jz wxfcb0 ; no
                       ; Set xfcb's password mode to sfcb's mode
27ED 70C3A127          mov m,b! jmp wxfcb1

                endif

                func104:                       ; Set current date and time

                if MPM
```

```
                              call get$stamp$add
                    else
27F1 21F4FB                   lxi h,stamp
                    endif
27F4 CD0E28                   call copy$stamp
27F7 36000EFFC3               mvi m,0! mvi c,0ffh! jmp timef

                    func105:                          ; Get current date and time



                    if MPM
                              call get$stamp$add
                    else
27FE 0E00CD4EFF               mvi c,0! call timef
2803 21F4FB                   lxi h,stamp
                    endif

2806 EB                       xchg
2807 CD0E28                   call copy$stamp
280A 1AC3720A                 ldax d! jmp sta$ret

                    copy$stamp:
280E 0E04C3290E               mvi c,4! jmp move ; ret

                    if MPM

                    get$stamp$add:
                              call rlradr! lxi b,-5! dad b
                              ret
                    endif

                    if BANKED

                    func106:                          ; Set default password

                    if MPM
                              call get$df$pwa! inr a! rz
                              lxi b,7! dad b
                    else
2813 21EE28                   lxi h,df$password+7
                    endif
2816 EB010800E5               xchg! lxi b,8! push h
281B C34E20                   jmp set$pw0
                    else

                    func106        equ     func$ret

                    endif

                    func107:                          ; Return serial number

                    if MPM
                              lhld sysdat! mvi l,181
                    else
281E 210000                   lxi h,serial
```

```
                          endif

 2821 EB0E06C329                 xchg! mvi c,6! jmp move

                          func108:                           ; Get/Set program return code

                                 ; Is de = 0ffffh?
 2827 7AA33C                     mov a,d! ana e! inr a
 282A 2AACFBCA29                 lhld clp$errcde! jz sthl$ret ; yes - return return code
 2830 EB22ACFBC9                 xchg! shld clp$errcde! ret ; no - set return code

                          goback0:
 2835 21FFFF229F                 lxi h,0ffffh! shld aret
                          goback:
                                 ; Arrive here at end of processing to return to user
 283B 3ADDFBB7CA                 lda resel! ora a! jz retmon

                          if MPM
                                          lda comp$fcb$cks! ora a! cnz set$chksum$fcb
                          endif

 2842 2ADBFB3ABF                          lhld info! lda fcbdsk! mov m,a ; fcb(0)=fcbdsk
                          if BANKED

                                          ; fcb(7) = fcb(7) | xfcb$read$only
 2849 110700193A                          lxi d,7! dad d! lda xfcb$read$only! ora m! mov m,a

                          endif
                          if MPM
                                          ; if high$ext = 60h then fcb(8) = fcb(8) | 80h
                                          ;                     else fcb(ext) = fcb(ext) | high$ext

                                          call getexta! lda high$ext! cpi 60h! jnz goback2
                                          lxi d,-4! dad d! mvi a,80h
                                 goback2:
                                          ora m! mov m,a
                          else
                                          ; fcb(8) = fcb(8) | high$ext
                          if BANKED
 2852 23                                  inx h
                          else
                                          lxi d,8! dad d
                          endif
 2853 3AB528B677                          lda high$ext! ora m! mov m,a
                          endif

                          ;      return from the disk monitor

                          retmon:
 2858 2A3203F9                   lhld entsp! sphl
 285C 2A9F0D7D44                 lhld aret! mov a,l! mov b,h! ret
                          ;
                          ;      data areas
                          ;
 2862 E5                  efcb:  db      empty    ; 0e5=available dir entry
 2863 0000                rodsk: dw      0        ; read only disk vector
```

```
2865 0000       dlog:  dw       0       ; logged-in disks

                if MPM

                rlog:  dw       0       ; removeable logged-in disks
                tlog:  dw       0       ; removeable disk test login vector
                ntlog: dw       0       ; new tlog vector
                rem$drv: ds     byte    ; curdsk removable drive switch
                                        ; 0 = permanent drive, 1 = removable drive
                endif

                if not BANKED

                xdmaad equ      $
                curdma ds       word    ; current dma address

                endif

                if not MPM

2867            buffa: ds       word    ; pointer to directory dma address

                endif

                ;
                ;       curtrka - alloca are set upon disk select
                ;       (data must be adjacent, do not insert variables)
                ;       (address of translate vector, not used)
2869            cdrmaxa:ds      word    ; pointer to cur dir max value (2 bytes)
286B            curtrka:ds      word    ; current track address (2)
286D            curreca:ds      word    ; current record address (3)
286F            drvlbla:ds      word    ; current drive label byte address (1)
2871            lsn$add:ds      word    ; login sequence # address (1)
                                        ; +1 -> bios media change flag (1)
2873            dpbaddr:ds      word    ; current disk parameter block address
2875            checka:         ds      word    ; current checksum vector address
2877            alloca:         ds      word    ; current allocation vector address
2879            dirbcba:ds      word    ; dir bcb list head
287B            dtabcba:ds      word    ; data bcb list head
                hash$tbla:
287D                    ds      word    ; directory hash table address
287F                    ds      byte    ; directory hash table bank

000D =          addlist         equ     $-dpbaddr       ; address list size


                ;
                ;               buffer control block format
                ;
                ; bcb format : drv(1) || rec(3) || pend(1) || sequence(1) ||
                ;               0        1         4          5
                ;
                ;               track(2) || sector(2) || buffer$add(2) ||
                ;               6           8            10
                ;
                ;               bank(1) || link(2)
                ;               12         13
                ;

                ;       sectpt - offset obtained from disk parm block at dpbaddr
                ;       (data must be adjacent, do not insert variables)
2880            sectpt:         ds      word    ; sectors per track
2882            blkshf:         ds      byte    ; block shift factor
2883            blkmsk:         ds      byte    ; block mask
2884            extmsk:         ds      byte    ; extent mask
2885            maxall:         ds      word    ; maximum allocation number
2887            dirmax:         ds      word    ; largest directory number
2889            dirblk:         ds      word    ; reserved allocation bits for directory
288B            chksiz:         ds      word    ; size of checksum vector
288D            offset:         ds      word    ; offset tracks at beginning
288F            physhf:         ds      byte    ; physical record shift
2890            phymsk:         ds      byte    ; physical record mask
0011 =          dpblist         equ     $-sectpt        ; size of area
                ;
                ;       local variables
                ;
2891            drec            ds      word    ; directory record number
2893            blk$off:        ds      byte    ; record offset within block
2894            last$off:       ds      byte    ; last offset within new block
2895            last$drive:     ds      byte    ; drive of last new block
2896            last$block:     ds      word    ; last new block

                ; The following two variables are initialized as a pair on entry

2898            dir$cnt:        ds      byte    ; direct i/o count
2899            mult$num:       ds      byte    ; multi-sector number

289A            tranv: ds       word    ; address of translate vector
                lock$unlock:
                make$flag:
289C            rmf:   ds       byte    ; read mode flag for open$reel
                incr$pdcnt:
289D            dirloc:         ds      byte    ; directory flag in rename, etc.
                free$mode:
289E            linfo: ds       byte    ; low(info)
289F            dminx: ds       byte    ; local for diskwrite

                if MPM

                searchl:ds      byte    ; search length

                endif
                if BANKED

28A0            searcha:ds      word    ; search address

                endif

                if BANKED

                save$xfcb:
28A2                    ds      byte    ; search xfcb save flag

```

```
                endif

28A3            single:         ds      byte    ; set true if single byte allocation map

                if MPM

                seldsk: ds      byte    ; currently selected disk

                endif

28A4            seldsk:         ds      byte    ; disk on entry to bdos
28A5            rcount:         ds      byte    ; record count in current fcb
28A6            extval:         ds      byte    ; extent number and extmsk
                save$mod:
28A7                    ds      byte    ; open$reel module save field

28A8            vrecord:ds      byte    ; current virtual record

                if not MPM

28A9 FF         curdsk: db      0ffh    ; current disk

                endif

28AA FF         adrive: db      0ffh    ; current blocking/deblocking disk
28AB            arecord:ds      word    ; current actual record
28AD                    ds      byte

28AE            save$ranr:      ds      3       ; random record save area
28B1            arecord1:       ds      word    ; current actual block# * blkmsk
28B3            attributes:     ds      byte    ; make attribute hold area
28B4            readf$sw:       ds      byte    ; BIOS read/write switch

                ;******** following variable order critical ****************

                if MPM

                mult$cnt:       ds      byte    ; multi-sector count
                pdcnt:          ds      byte    ; process descriptor count

                endif

28B5            high$ext:       ds      byte    ; fcb high ext bits

                if BANKED

28B6            xfcb$read$only:         ds      byte    ; xfcb read only flag

                endif
                if MPM

                curdsk:                 db      0ffh                    ;current disk
                packed$dcnt:    ds      3                       ;
                pdaddr:                 ds      word                    ;
                ;************************************************************
                cur$pos:        ds      word                    ;
                prv$pos:        ds      word                    ;
                sdcnt:          ds      word                    ;
                sdblk:          ds      word                    ;
                sdcnt0:                 ds      word                    ;
                sdblk0:                 ds      word                    ;
                dont$close:     ds      byte                    ;
                open$cnt:                       ; mp/m temp variable for open
                lock$cnt:       ds      word    ; mp/m temp variable for lock
                file$id:        ds      word    ; mp/m temp variable for lock
                deleted$files:  ds      byte
                lock$shell:     ds      byte
                lock$sp:        ds      word
                set$ro$flag:    ds      byte
                check$disk:     ds      byte
                flushed:        ds      byte
                fcb$cks$valid:  ds      byte
                ;                               mp/m variables *

                endif

                ;       local variables for directory access
        28B7    dptr:   ds      byte    ; directory pointer 0,1,2,3

        28B8    save$hash:      ds      4       ; hash code save area

                if BANKED

        28BC    copy$cr$init:   ds      byte    ; copy$cr$only initialization value

                else

                hashmx:         ds      word    ; cdrmax or dirmax
                xdcnt:  ds      word    ; empty directory dcnt

                endif

                if MPM

                xdcnt:  ds      word    ; empty directory dcnt
                xdblk:  ds      word    ; empty directory block
                dcnt:   ds      word    ; directory counter 0,1,...,dirmax
                dblk:   ds      word    ; directory block index

                endif

        28BD    search$user0:   ds      byte    ; search user 0 for file (open)

        28BE    user0$pass:     ds      byte    ; search user 0 pass flag

        28BF    fcbdsk:                 ds      byte    ; disk named in fcb

                if MPM

                make$xfcb:      ds      1
                find$xfcb:      ds      1

```

```
                        endif

28C0 0F10111316log$fxs:db       15,16,17,19,22,23,30,35,99,100,102,103,0
28CD 1415212228rw$fxs: db       20,21,33,34,40,41,0
28D4 101200     sc$fxs: db      16,18,0

                if MPM

                comp$fcb$cks:   ds      byte    ; compute fcb checksum flag

                endif
                if BANKED

28D7            pw$fcb:                 ds      12      ;1 |
28E3 00                         db      0       ;2 |
28E4 00         pw$mode:        db      0       ;3 |- Order critical
28E5 00                         db      0       ;4 |
28E6 00                         db      0       ;5 |

28E7            df$password:    ds      8

                if MPM
                                ds      120
                endif
                endif

28EF            phy$off:        ds      byte
28F0            curbcba:        ds      word

                if BANKED

28F2            lastbcba:       ds      word
28F4            rootbcba:       ds      word
28F6            emptybcba:      ds      word
28F8            seqbcba:        ds      word
28FA            buffer$bank:    ds      byte

                endif

28FB            track:          ds      word
28FD            sector:                 ds      word

                ;       **************************
                ;       Blocking/Deblocking Module
                ;       **************************

                deblock$dta:
28FF 2A7B28             lhld dtabcba

                if BANKED
2902 FE04C25829         cpi 4! jnz deblock
                deblock$flush:
                        ; de = addr of 1st bcb
2907 5E2356             mov e,m! inx h! mov d,m
                        ; Search for dirty bcb with lowest track #
290A 21FFFF22FB         lxi h,0ffffh! shld track! xchg
```

```
                    deblock$flush1:
                            ; Does current drive own bcb?
2911 3AAA28BEC2             lda adrive! cmp m! jnz deblock$flush2 ;no
                            ; Is bcb's buffer pending?
2918 EB21040019             xchg! lxi h,4! dad d! mov a,m
291E EB3CC23C29             xchg! inr a! jnz deblock$flush2 ; no
                            ; Is bcb(6) < track?
2923 E51313EB               push h! inx d! inx d! xchg
2927 5E2356                 mov e,m! inx h! mov d,m
                            ; Subdh computes hl = de - hl
292A 2AFB28CD7F             lhld track! call subdh! pop h! jnc deblock$flush2 ; no
                            ; yes - track = bcb(6) , sector = addr(bcb)
2934 EB22FB28EB             xchg! shld track! xchg! shld sector
                    deblock$flush2:
                            ; Is this the last bcb?
293C CD782BC211             call get$next$bcba! jnz deblock$flush1 ; no - hl = addr of next bcb
                            ; Does track = ffff?
2942 21FB28CDFB             lxi h,track! call test$ffff! rz ; yes - no bcb to flush
                            ; Flush bcb located by sector
2949 2AFD28AF3E             lhld sector! xra a! mvi a,4! call deblock
2952 2A7B28C307             lhld dtabcba! jmp deblock$flush ; Repeat until no bcb's to flush
                    endif

                    deblock:

                            ; BDOS Blocking/Deblocking routine
                            ; a = 1 -> read command
                            ; a = 2 -> write command
                            ; a = 3 -> locate command
                            ; a = 4 -> flush command
                            ; a = 5 -> directory update

2958 F5                     push a ; Save z flag and deblock fx

                            ; phy$off = low(arecord) & phymsk
                            ; low(arecord) = low(arecord) & ~phymsk
2959 CD512A                 call deblock8
295C 3AAB285FA0             lda arecord! mov e,a! ana b! sta phy$off
2964 7BA132AB28             mov a,e! ana c! sta arecord

                    if BANKED
2969 F1F5C4922A             pop a! push a! cnz get$bcba
                    endif

296E 22F028CD82             shld curbcba! call getbuffa! shld curdma
                            ; hl = curbcba, de = .adrive, c = 4
2977 CD582A                 call deblock9
                            ; Is BCB discarded?
297A 7E3CCAC229             mov a,m! inr a! jz deblock2 ; yes
                            ; Is command flush?
297F F1F5FE04D2             pop a! push a! cpi 4! jnc deblock1 ; yes
                            ; Is referenced physical record already in buffer?
2986 CD1F0ECAF2             call compare! jz deblock45 ; yes
298C AF                     xra a
                    deblock1:
                            ; Does buffer contain an updated record?
```

```
298D CD612A                     call deblock10
2990 FE05CA9A29                 cpi 5! jz deblock15
2995 7EB7CAC229                 mov a,m! ora a! jz deblock2 ; no
                        deblock15:
                                ; Reset record pending flag
299A 3600                       mvi m,0
                                ; Save arecord
299C 2AAB28E53A                 lhld arecord! push h! lda arecord+2! push a
                                ; Flush physical record buffer
29A4 CD582A                     call deblock9
29A7 EBCD290E                   xchg! call move
                                ; Select drive to be flushed
29AB 21A9283AAA                 lxi h,curdsk! lda adrive! cmp m! cnz disk$select1
                                ; Write record if drive logged-in
29B5 3E01CC712A                 mvi a,1! cz deblock$io
                                ; Restore arecord
29BA C1D1CD030F                 pop b! pop d! call set$arecord
                                ; Restore selected drive
29BF CD6B1E                     call curselect
                        deblock2:
                                ; Is deblock command flush | dir write?
29C2 F1FE04D0                   pop a! cpi 4! rnc ; yes - return
                                ; Is deblock command write?
29C6 F5FE02C2D6                 push a! cpi 2! jnz deblock25 ; no
                                ; Is blk$off < last$off
29CC 2194283A93                 lxi h,last$off! lda blk$off! cmp m! jnc deblock3 ; no
                        deblock25:
                                ; Discard BCB on read operations in case
                                ; I/O error occurs
29D6 2AF02836FF                 lhld curbcba! mvi m,0ffh
                                ; Read physical record buffer
29DB 3E02C3E329                 mvi a,2! jmp deblock35
                        deblock3:
                                ; last$off = blk$off + 1
29E0 3C77                       inr a! mov m,a
                                ; Place track & sector in bcb
29E2 AF                         xra a
                        deblock35:
29E3 CD712A                     call deblock$io
                        deblock4:
29E6 CD582A                     call deblock9 ; phypfx = adrive || arecord
29E9 CD290E3600                 call move! mvi m,0 ; zero pending flag

                        if BANKED
                                ; Zero logical record sequence
29EE 23CD832B                   inx h! call set$bcb$seq
                        endif

                        deblock45:
                                ; recadd = phybuffa + phy$off*80h
29F2 3AEF283C11                 lda phy$off! inr a! lxi d,80h! lxi h,0ff80h
                        deblock5:
29FC 193DC2FC29                 dad d! dcr a! jnz deblock5
2A01 EB2A18FD19                 xchg! lhld curdma! dad d
                                ; If deblock command = locate then buffa = recadd; return
2A06 F1FE03C210                 pop a! cpi 3! jnz deblock6
```

CP/M PLUS V3.0

SER. # CP3-135

```
2A0C 226728C9           shld buffa! ret
                 deblock6:
2A10 EB2AD8FB01         xchg! lhld dmaad! lxi b,80h
                        ; If deblock command = read
2A17 FE01               cpi 1

                 if BANKED
2A19 C2332A             jnz deblock7
                        ; then move to tpa
2A1C 3AFAFB3DBA         lda common$base+1! dcr a! cmp d! jc move$tpa
2A24 3AFA284F06         lda buffer$bank! mov c,a! mvi b,1! call deblock12
2A2D 018000C30C         lxi b,80h! jmp move$tpa
                 deblock7:

                 else
                        jz move$tpa ; then move to dma
                 endif

                        ; else move from dma
2A33 EB                 xchg

                 if BANKED
2A34 3AFAFB3DBC         lda common$base+1! dcr a! cmp h! jc deblock75
2A3C 3AFA28470E         lda buffer$bank! mov b,a! mvi c,1! call deblock12
2A45 018000             lxi b,80h
                 deblock75:

                 endif

2A48 CD0CFD             call move$tpa
                        ; Set physical record pending flag for write command
2A4B CD612A36FF         call deblock10! mvi m,0ffh
2A50 C9                 ret

                 deblock8:
2A51 3A9028472F         lda phymsk! mov b,a! cma! mov c,a! ret

                 deblock9:
2A58 2AF02811AA         lhld curbcba! lxi d,adrive! mvi c,4! ret

                 deblock10:
2A61 110400             lxi d,4
                 deblock11:
2A64 2AF02819C9         lhld curbcba! dad d! ret

                 if BANKED

                 deblock12:
2A69 E5D5CD57FF         push h! push d! call xmovef
2A6E D1E1C9             pop d! pop h! ret
                 endif

                 deblock$io:
                        ; a = 0 -> seek only
                        ; a = 1 -> write
                        ; a = 2 -> read
```

```
2A71 F5CD0C0F           push a! call seek

                if BANKED
2A75 3AFA28CD54         lda buffer$bank! call setbnkf
                endif

2A7B 0E01               mvi c,1
2A7D F13D               pop a! dcr a
2A7F CA9E0E             jz wrbuff
2A82 F4930E             cp rdbuff
                        ; Move track & sector to bcb
2A85 CD612A2323         call deblock10! inx h! inx h
2A8A 11FB2B0E04         lxi d,track! mvi c,4! jmp move

                if BANKED

                get$bcba:
2A92 22F428             shld rootbcba
2A95 11F3FF1922         lxi d,-13! dad d! shld lastbcba
2A9C CD782BE5           call get$next$bcba! push h
                        ; Is there only 1 bcb in list?
2AA0 CD782BE1C8         call get$next$bcba! pop h! rz ; yes - return
2AA5 EB21000022         xchg! lxi h,0! shld emptybcba! shld seqbcba
2AAF EB                 xchg
                get$bcb1:
                        ; Does bcb contain requested record?
2AB0 22F028CD58         shld curbcba! call deblock9! call compare! jz get$bcb4 ; yes
                        ; Is bcb discarded?
2ABC 2AF0287E3C         lhld curbcba! mov a,m! inr a! jnz get$bcb11 ; no
2AC4 EB2AF22822         xchg! lhld lastbcba! shld emptybcba! jmp get$bcb14
                get$bcb11:
                        ; Does bcb contain record from current disk?
2ACE 3AAA28BEC2         lda adrive! cmp m! jnz get$bcb15 ; no
2AD5 EB21050019         xchg! lxi h,5! dad d! lda phy$msk
                        ; Is phy$msk = 0?
2ADD B7CAF32A           ora a! jz get$bcb14 ; yes
                        ; Does bcb(5) [bcb sequence] = phymsk?
2AE1 BEC2F32A           cmp m! jnz get$bcb14 ; no
2AE5 2AF8287DB4         lhld seqbcba! mov a,l! ora h! jnz get$bcb14
2AED 2AF22822F8         lhld lastbcba! shld seqbcba
                get$bcb14:
2AF3 EB                 xchg
                get$bcb15:
                        ; Advance to next bcb - list exhausted?
2AF4 E5CD782BD1         push h! call get$next$bcba! pop d! jz get$bcb2 ; yes
2AFC EB22F228EB         xchg! shld lastbcba! xchg! jmp get$bcb1
                get$bcb2:
                        ; Matching bcb not found
                        ; Was a sequentially accessed bcb encountered?
2B04 2AF8287DB4         lhld seqbcba! mov a,l! ora h! jnz get$bcb25 ; yes
                        ; Was a discarded bcb encountered?
2B0C 2AF6287DB4         lhld emptybcba! mov a,l! ora h! jz get$bcb3 ; no
                get$bcb25:
2B14 22F228             shld lastbcba
                get$bcb3:
                        ; Insert selected bcb at head of list
```

```
2B17 2AF228CD78       lhld lastbcba! call get$next$bcba
2B1D 22F028CD78       shld curbcba! call get$next$bcba
2B23 EBCD6D2B         xchg! call last$bcb$links$de
2B27 2AF4285E23       lhld rootbcba! mov e,m! inx h! mov d,m
2B2D 2AF028010D       lhld curbcba! lxi b,13! dad b
2B34 732372           mov m,e! inx h! mov m,d
2B37 2AF028EB2A       lhld curbcba! xchg! lhld rootbcba
2B3E 732372EBC9       mov m,e! inx h! mov m,d! xchg! ret
                get$bcb4:
                      ; BCB matched arecord
2B43 2AF0281105       lhld curbcba! lxi d,5! dad d
                      ; Does bcb(5) = phy$off?
2B4A 3AEF28BECA       lda phy$off! cmp m! jz get$bcb5 ; yes
                      ; Does bcb(5) + 1 = phy$off?
2B51 34BECA592B       inr m! cmp m! jz get$bcb5 ; yes
2B56 CD832B           call set$bcb$seq
                get$bcb5:
                      ; Is bcb at head of list?
2B59 2AF028EB2A       lhld curbcba! xchg! lhld rootbcba
2B60 7E236E67         mov a,m! inx h! mov l,m! mov h,a
2B64 CD7F11B5EB       call subdh! ora l! xchg! rz ; yes
2B6A C3172B           jmp get$bcb3 ; no - insert bcb at head of list

                last$bcb$links$de:
2B6D 2AF228010D       lhld lastbcba! lxi b,13! dad b
2B74 732372C9         mov m,e! inx h! mov m,d! ret

                get$next$bcba:
2B78 010D00095E       lxi b,13! dad b! mov e,m! inx h! mov d,m
2B7F EB7CB5C9         xchg! mov a,h! ora l! ret

                set$bcb$seq:
2B83 3AEF2877B7       lda phy$off! mov m,a! ora a! rz
2B89 3A90283C77       lda phy$msk! inr a! mov m,a! ret

                endif

                if not MPM
                if not BANKED

                            ds      112
                last:
                            org     base + (((last-base)+255) and 0ff00h) - 112

                olog:       dw      0
                rlog:       dw      0

                patch$flgs: dw      0,0
                            dw      base+6
                            xra a! ret

                ; System Control Block

                SCB:

                ; Expansion Area - 6 bytes
```

```

hashl:		db	0
hash:		dw	0,0
version:	db	31h

; Utilities Section - 8 bytes

util$flgs:	dw	0,0
dspl$flgs:	dw	0
		dw	0

; CLP Section - 4 bytes

clp$flgs:	dw	0
clp$errcde:	dw	0

; CCP Section - 8 bytes

ccp$comlen:	db	0
ccp$curdrv:	db	0
ccp$curusr:	db	0
ccp$conbuff:	dw	0
ccp$flgs:	dw	0
		db	0

; Device I/O Section - 32 bytes

conwidth:	db	0
column:			db	0
conpage:	db	0
conline:	db	0
conbuffadd:	dw	0
conbufflen:	dw	0
conin$rflg:	dw	0
conout$rflg:	dw	0
auxin$rflg:	dw	0
auxout$rflg:	dw	0
lstout$rflg:	dw	0
page$mode:	db	0
pm$default:	db	0
ctlh$act:	db	0
rubout$act:	db	0
type$ahead:	db	0
contran:	dw	0
conmode:	dw	0
		db	0
		db	0
outdelim:	db	'$'
listcp		db	0
qflag:		db	0

; BDOS Section - 42 bytes

scbadd:			dw	scb
dmaad:		dw	0080h
olddsk:			db	0
```

```
                        info:           dw      0
                        resel:          db      0
                        relog:          db      0
                        fx:             db      0
                        usrcode:        db      0
                        dcnt:           dw      0
                        searcha:        dw      0
                        searchl:        db      0
                        multcnt:        db      1
                        errormode:      db      0
                        searchchain:    db      0,0ffh,0ffh,0ffh
                        temp$drive:     db      0
                        errdrv:                 db      0
                                        dw      0
                        media$flag:     db      0
                                        dw      0
                        bdos$flags:     db      0
                        stamp:          db      0ffh,0ffh,0ffh,0ffh,0ffh
                        commonbase:     dw      0
                        error:                  jmp     error$sub
                        bdosadd:        dw      base+6

                        endif
                        endif

                        ;       ************************
                        ;       Directory Hashing Module
                        ;       ************************

                        ; Hash format
                        ; xxsuuuuu xxxxxxxx xxxxxxxx ssssssss
                        ; x = hash code of fcb name field
                        ; u = low 5 bits of fcb user field
                        ;     1st bit is on for XFCB's
                        ; s = shiftr(mod || ext,extshf)

                        if not BANKED

                        hashorg:
                                org     base+(((hashorg-base)+255) and 0ff00h)
                        endif

                        init$hash:
                                ; de = .hash table entry
                                ; hl = .dir fcb
2B8F E5D5CDE02B                 push h! push d! call get$hash
                                ; Move computed hash to hash table entry
2B94 E1119DFB01                 pop h! lxi d,hash! lxi b,4

                        if BANKED
2B9B 3A7F28CD09                 lda hash$tbla+2! call move$out
                        else
                                call movef
                        endif

                                ; Save next hash table entry address
```

```
 7372  2BA1 22B128             shld arecord1
 7373                          ; Restore dir fcb address
 7374  2BA4 E1C9               pop h! ret
 7375
 7376                  set$hash:
 7377                          ; Return if searchl = 0
 7378  2BA6 B7C8               ora a! rz
 7379                          ; Is searchl < 12 ?
 7380  2BA8 FE0CDADC2B         cpi 12! jc set$hash2 ; yes - hashl = 0
 7381                          ; Is searchl = 12 ?
 7382  2BAD 3E02CAB42B         mvi a,2! jz set$hash1 ; yes - hashl = 2
 7383  2BB2 3E03               mvi a,3 ; hashl = 3
 7384                  set$hash1:
 7385  2BB4 329CFB             sta hashl
 7386  2BB7 EB                 xchg
 7387                          ; Is dir hashing invoked for drive?
 7388  2BB8 CD642CC8           call test$hash! rz ; no
 7389  2BBC EB                 xchg
 7390  2BBD 3ADFFB             lda fx
 7391  2BC0 FE10CAE02B         cpi 16! jz get$hash ; bdos fx = 16
 7392  2BC5 FE23CACF2B         cpi 35! jz set$hash15
 7393  2BCA FE14D2E02B         cpi 20! jnc get$hash ; bdos fx = 20 or above
 7394                  set$hash15:
 7395  2BCF 3E02329CFB         mvi a,2! sta hashl ; bdos fx = 15,17,18,19, or 35
 7396                          ; if fcb wild then hashl = 0, hash = fcb(0)
 7397                          ;              else hashl = 2, hash = get$hash
 7398  2BD4 E5CD3917E1         push h! call chk$wild! pop h! jnz get$hash
 7399                  set$hash2:
 7400  2BDC AF329CFB           xra a! sta hashl
 7401                          ; jmp get$hash
 7402
 7403                  get$hash:
 7404                          ; hash(0) = fcb(0)
 7405  2BE0 7E329DFB23         mov a,m! sta hash! inx h! xchg
 7406                          ; Don't compute hash for dir lbl & sfcb's
 7407  2BE6 210000E620         lxi h,0! ani 20h! jnz get$hash6
 7408                          ; b = 11, c = 8, ahl = 0
 7409                          ; Compute fcb name hash (000000xx xxxxxxxx xxxxxxxx) (ahl)
 7410  2BEE 01080B             lxi b,0b08h
 7411                  get$hash1:
 7412                          ; Don't shift if fcb(8)
 7413  2BF1 0DC5CA062C         dcr c! push b! jz get$hash3
 7414                          ; Don't shift if fcb(6)
 7415  2BF6 0D0DCA062C         dcr c! dcr c! jz get$hash3
 7416                          ; ahl = ahl * 2
 7417  2BFB 298FF578           dad h! adc a! push a! mov a,b
 7418                          ; is b odd?
 7419  2BFF 1FDA072C           rar! jc get$hash4 ; yes
 7420                          ; ahl = ahl * 2 for even fcb(i)
 7421  2C03 F1298F             pop a! dad h! adc a
 7422                  get$hash3:
 7423  2C06 F5                 push a
 7424                  get$hash4:
 7425                          ; a = fcb(i) & 7fh - 20h divided by 2 if even
 7426  2C07 1AE67FD620         ldax d! ani 7fh! sui 20h! rar! jnc get$hash5
 7427  2C10 17                 ral
```

```
                     get$hash5:
                             ; ahl = ahl + a
2C11 4F0600                  mov c,a! mvi b,0
2C14 F109CE00C1              pop a! dad b! aci 0! pop b
                             ; advance to next fcb char
2C19 1305C2F12B              inx d! dcr b! jnz get$hash1
                     get$hash6:
                             ; ahl = 000000xx xxxxxxxx xxxxxxxx
                             ; Store low 2 bytes of hash
2C1E 229EFB219D              shld hash+1! lxi h,hash
                             ; hash(0) = hash(0) (000uuuuu) | xx000000
2C24 E6030F0FB6              ani 3! rrc! rrc! ora m! mov m,a
                             ; Does fcb(0) = e5h, 20h, or 21h?
2C2A E620C25B2C              ani 20h! jnz get$hash9 ; yes
                             ; bc = 00000mmm mmmeeeee, m = module #, e = extent
2C2F 1AE61F4F13              ldax d! ani 1fh! mov c,a! inx d! inx d
2C35 1AE63F0F0F              ldax d! ani 3fh! rrc! rrc! rrc! mov d,a
2C3C E607477AE6              ani 7! mov b,a! mov a,d! ani 0e0h! ora c! mov c,a
                             ; shift bc right by # of bits in extmsk
2C44 3A8428                  lda extmsk
                     get$hash7:
2C47 1FD2562C                rar! jnc get$hash8
2C4B F5                      push a
2C4C 781F47                  mov a,b! rar! mov b,a
2C4F 791F4F                  mov a,c! rar! mov c,a
2C52 F1C3472C                pop a! jmp get$hash7
                     get$hash8:
                             ; hash(0) = hash(0) (xx0uuuuu) | 00s00000
2C56 78E6010F0F              mov a,b! ani 1! rrc! rrc
                     get$hash9:
2C5B 0FB677                  rrc! ora m! mov m,a
                             ; hash(3) = ssssssss
2C5E 1103001971              lxi d,3! dad d! mov m,c! ret

                     test$hash:
2C64 2A7D287DB4              lhld hash$tbla! mov a,l! ora h! inr a! ret

                     search$hash:
                             ; Does hash table exist for drive?
2C6B CD642CC8                call test$hash! rz ; no
                             ; Has dir hash search been disabled?
2C6F 3A9CFB3CC8              lda hashl! inr a! rz ; yes
                             ; Is searchl = 0?
2C74 3AE5FBB7C8              lda searchl! ora a! rz ; yes
                             ; hashmx = cdrmaxa if searchl ^= 1
                             ;          dir$max if searchl = 1
2C79 2A69285E23              lhld cdrmaxa! mov e,m! inx h! mov d,m
2C7F EB3DC2872C              xchg! dcr a! jnz search$h0
2C84 2A8728                  lhld dir$max
                     search$h0:
2C87 2212FD                  shld hashmx

                     if BANKED
                             ; call search$hash in resbdos, a = bank, hl = hash tbl addr
2C8A 3A7F282A7D              lda hash$tbla+2! lhld hash$tbla! call srch$hash
                             ; Was search successful?
```

```
2C93 C2AE2C           jnz search$h1 ; no
                      ; Is directory read required?
2C96 3A14FDB70E       lda rd$dir$flag! ora a! mvi c,0
2C9C C43D13           cnz r$dir2 ; yes if Z flag reset
                      ; Is function = 18?
2C9F 3ADFFBD612       lda fx! sui 18! rz ; Never reset dcnt for fx 18
                      ; Was media change detected by above read?
2CA5 3A9CFB3CCC       lda hashl! inr a! cz setenddir ; yes
2CAC AFC9             xra a! ret ; search$hash successful
                search$h1:
                      ; Was search initiated from beginning of directory?
2CAE CDF812C0         call end$of$dir! rnz ; no
                      ; Is bdos fx = 15,17,19,22,23,30?
2CB2 CDF511C0         call tst$log$fxs! rnz ; no
                      ; Disable hash & return successful
2CB6 3EFF329CFB       mvi a,0ffh! sta hashl
2CBB 2A69285E23       lhld cdrmaxa! mov e,m! inx h! mov d,m! xchg
2CC2 2BCD0D19AF       dcx h! call set$dcnt$dblK1! xra a! ret
                else
                      lhld hash$tbla! mov b,h! mov c,l
                      lhld hashmx! xchg
                      ; Return with Z flag set if dcnt = hashmx
                      lhld dcnt! push h! call subdh! pop d! ora l! rz
                      ; Push hashmx - dcnt (# of hashtbl entries to search)
                      ; Push dcnt + 1
                      push h! inx d! xchg! push h
                      ; Compute .hash$tbl(dcnt)
                      dcx h! dad h! dad h! dad b
                search$h1:
                      ; Advance hl to address of next hash$tbl entry
                      lxi d,4! dad d! lxi d,hash
                      ; Do hash u fields match?
                      ldax d! xra m! ani 1fh! jnz search$h3 ; no
                      ; Do hash's match?
                      call search$h6! jz search$h4 ; yes
                search$h2:
                      xchg! pop h
                search$h25:
                      ; de = .hash$tbl(dcnt), hl = dcnt
                      ; dcnt = dcnt + 1
                      inx h! xthl
                      ; hl = # of hash$tbl entries to search
                      ; decrement & test for zero
                      ; Restore stack & hl to .hashtbl(dcnt)
                      dcx h! mov a,l! ora h! xthl! push h
                      ; Are we done?
                      xchg! jnz search$h1 ; no - Keep searching
                      ; Search unsuccessful
                      pop h! pop h
                      ; Was search initiated from beginning of directory?
                      call end$of$dir! rnz ; no
                      ; Is fx = 15,17,19,22,23,30 & drive removeable?
                      call tst$log$fxs! rnz ; no
                      ; Disable hash & return successful
                      mvi a,0ffh! sta hashl
                      lhld cdrmaxa! mov e,m! inx h! mov d,m! xchg
```

```
        dcx h! call set$dcnt$dblk1! xra a! ret

search$h3:
        ; Does xdcnt+1 = 0ffh?
        lda xdcnt+1! inr a! jz search$h5 ; yes
        ; Does xdcnt+1 = 0feh?
        inr a! jnz search$h2 ; no - continue searching
        ; Do hash's match?
        call search$h6! jnz search$h2 ; no
        ; xdcnt+1 = 0feh
        ; Open user 0 search
        ; Does hash u field = 0?
        mov a,m! ani 1fh! jnz search$h2 ; no
        ; Search successful
search$h4:
        ; Successful search
        ; Set dcnt to search$hash dcnt-1
        ; dcnt gets incremented by read$dir
        ; Also discard search$hash loop count
        lhld dcnt! xchg
        pop h! dcx h! shld dcnt! pop b
        ; Does dcnt&3 = 3?
        mov a,l! ani 03h! cpi 03h! rz ; yes
        ; Does old dcnt & new dcnt reside in same sector?
        mov a,e! ani 0fch! mov e,a
        mov a,l! ani 0fch! mov l,a
        call subdh! ora l! rz ; yes
        ; Read directory record
        call read$dir2
        ; Has media change been detected?
        lda hashl! inr a! cz setenddir ; dcnt = -1 if hashl = 0ffh
        xra a! ret
search$h5:
        ; xdcnt+1 = 0ffh
        ; Make search to save dcnt of empty fcb
        ; Is hash$tbl entry empty?
        mov a,m! cpi 0f5h! jnz search$h2 ; no
search$h55:
        ; xdcnt = dcnt
        xchg! pop h! shld xdcnt! jmp search$h25
search$h6:
        ; hash compare routine
        ; Is hashl = 0?
        lda hashl! ora a! rz ; yes - hash compare successful
        ; b = 0f0h if hashl = 3
        ;     0d0h if hashl = 2
        mov c,a! rrc! rrc! rrc! ori 1001$0000b! mov b,a
        ; hash s field must be screened out of hash(0)
        ; if hashl = 2
        ; Do hash(0) fields match?
        ldax d! xra m! ana b! rnz ; no
        ; Compare remainder of hash fields for hashl bytes
        push h! inx h! inx d! call compare
        pop h! ret
endif

```

```
                fix$hash:
2CC8 CD642CC8           call test$hash! rz
2CCC 21B828119D         lxi h,save$hash! lxi d,hash! lxi b,4
2CD5 E5D5C5CD4B         push h! push d! push b! call movef
2CDB 2A7D28E5           lhld hash$tbla! push h
2CDF CD4011CDE0         call get$dptra! call get$hash
2CE5 2AE1FB2929         lhld dcnt! dad h! dad h
2CEA D119               pop d! dad d
2CEC C1D1D5C5           pop b! pop d! push d! push b

                if BANKED
2CF0 3A7F28CD09         lda hash$tbla+2! call move$out
                else
                        call movef
                endif

2CF6 C1E1D1C34B         pop b! pop h! pop d! jmp movef

                if not MPM
                if BANKED

2CFC                    ds      1
                last:
2CFF                    org     (((last-base)+255) and 0ff00h) - 1
2CFF 00                 db      0
                endif

                else
                        ds      192
                last:
                        org     (((last-base)+255) and 0ff00h) - 192

                        ;       bnkbdos patch area

                        dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                        dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                        dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                        dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                        dw      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                        dw      0,0,0,0,0,0,0,0,0,0,0,0

                free$root:      dw      $-$
                open$root:      dw      0
                lock$root:      dw      0
                lock$max:       db      0
                open$max:       db      0

                ;       BIOS access table

                bios    equ     $               ; base of the bios jump table
                bootf   equ     bios            ; cold boot function
                wbootf  equ     bootf+3         ; warm boot function
                constf  equ     wbootf+3        ; console status function
                coninf  equ     constf+3        ; console input function
                conoutf         equ     coninf+3        ; console output function
                listf   equ     conoutf+3       ; list output function
```

```
 7652                punchf equ    listf+3        ; punch output function
 7653                readerf        equ     punchf+3       ; reader input function
 7654                homef  equ    readerf+3      ; disk home function
 7655                seldskf        equ     homef+3        ; select disk function
 7656                settrkf        equ     seldskf+3      ; set track function
 7657                setsecf        equ     settrkf+3      ; set sector function
 7658                setdmaf        equ     setsecf+3      ; set dma function
 7659                readf  equ    setdmaf+3      ; read disk function
 7660                writef equ    readf+3        ; write disk function
 7661                liststf        equ     writef+3       ; list status function
 7662                sectran        equ     liststf+3      ; sector translate
 7663
 7664                endif
 7665
 7666   2D00                        end
```

```
ADDH            1146  2374# 4151
ADDLIST         000D  1831  6743#
ADJUSTDMAAD     20A7  4819# 6479  6481
ADRIVE          28AA   595  1725  1898  2558  2569  4085  4144  4488  6836# 6982
                      7048  7130  7180
ADVDMA          037F   670   686#
ALLOCA          2877  2712  2744  2761  2762  2764  2854  2866  5985  6197  6736#
ALLOCWD         1C07  4149  4153#
APOS            0D28   993  1018  1056  1074  1091  1158  1159  1163  1167  1168
                      1202  1203  1211  1216  1225  1295  1323  1329  1431  1431
                      1603#
APOSI           0D25  1091  1225  1601#
ARECORD         28AB  1950  1961  1961  2089  2097  2098  2098  2107  4143  4173
                      4196  4204  4208  4235  5307  6837# 7017  7018  7043  7043
ARECORD1        28B1  2100  2874  2883  4194  6841# 7372
ARET            0D9F   244  1539  1656  1674# 1675  1717  4471  5262  5275  5398
                      5938  6018  6339  6656  6694
ATRAN           0FF2  2091# 3852  4095  4165  5264
ATTRIBUTES      28B3  3176  3203  3250  3374  3439  3456  5180  5501  5574  5650
                      5672  5692  5714  5794  5827  5882  5921  6842#
AUXINRFLG       FBC2   151# 7295#
AUXINSTF        FF36    95# 1492
AUXOUTRFLG      FBC4   152# 7296#
AUXOUTSTF       FF39    96# 1496
BACKONE         09C5  1421  1422  1427#
BACKSP          093D  1358# 1362
BACKUP          0557   948# 1302  1360
BACKX           08A8  1291# 1303
BACKX1          08BD  1294  1297#
BADSEEK         1DA9  4368  4372  4397#
BANKED          FFFF    22#   65   108   230   249   291   598   606   659   764
                       781   791   796   801   805   816   820   919   936   952
                       992   999  1012  1116  1140  1153  1252  1268  1272  1293
                      1308  1320  1363  1377  1390  1395  1411  1417  1442  1446
                      1512  1526  1585  1593  1619  1640  1751  1884  1910  2306
                      2554  2578  2599  2738  2756  2841  3015  3070  3149  3175
                      3193  3208  3219  3241  3259  3272  3380  3398  3405  3429
                      3441  3692  3721  3838  3860  3932  4045  4112  4177  4199
                      4226  4277  4390  4404  4542  4553  4582  4662  4676  4687
                      4693  4872  5589  5765  5793  5846  5877  6300  6322  6338
                      6357  6439  6508  6517  6526  6620  6666  6682  6711  6801
                      6807  6856  6893  6935  6953  6973  7020  7079  7098  7113
                      7137  7150  7162  7242  7352  7365  7480  7606  7615
BASE            0000    51#   55    56    57    58  7246  7246  7252  7336  7355
                      7355  7619  7626
BDEEBDEMHL      0DE4  1761# 1966  1979
BDEEBDEPHL      0DED  1766# 1943  1970
BDOSADD         FBFE   188# 2312  7336#
BDOSE           0006   223#
BDOSE1          002E   238#  241
BDOSE2          003B   226   236   239   242#  684
BDOSFLAGS       FBF3   184#  607  2739  2757  7332#
BDOSJMP         006F   257   264# 1721
BIOS            FF00    62# 6317  7645# 7646
BIOSPG          FF00    58#   62    63    75    76    77    78    79    83    84
                        85    86    87    88    89    90    91    92    93    94
                        95    96    97    98    99   100   101   102   103   104
                       105   106
BLKMSK          2883  1926  2104  3916  3967  4206  4236  4255  4947  6762#
BLKOFF          2893  2106  3885  6776# 7061
BLKOUT          0A9E  1566# 1575
BLKOUT1         0AB2  1569  1571#
BLKOUT2         0AB5  1570  1573#
BLKSHF          2882  1927  2043  2096  2952  4943  4949  6214  6761#
BLOCKOK         1BDE  4133  4136#
BNKBDOSPG       FC00    55#
BOOTF           FF00    63# 7646# 7647
BUFFA           2867  2194  2211  2373  2622  4885  6720# 7092
BUFFERBANK      28FA  2579  6959# 7102  7115  7151
BUFFERFULL      09E0  1412  1437#
BUFFNZERO       188C  3579  3582#
BYTE            0001  1690# 6707  6741  6761  6762  6763  6769  6770  6776  6777
                      6778  6783  6784  6789  6791  6793  6794  6798  6810  6814
                      6818  6822  6823  6824  6826  6828  6838  6842  6843  6849
                      6850  6854  6858  6873  6877  6878  6880  6881  6882  6883
                      6889  6895  6913  6915  6917  6932  6950  6959
CALLOPEN        21B9  5502  5505  5512#
CBDOS           0374   667   681#
CBDOS1          0378   683#
CBS1            0642  1057# 1059
CBS2            064C  1058  1060#
CBUFF           0AC1  1033  1597#
CBUFFLEN        0AC0  1035  1104  1596#
CCPCOMLEN       FBAE   135# 7278#
CCPCONBUFF      FBB1   138# 7281#
CCPCURDRV       FBAF   136# 7279#
CCPCURUSR       FBB0   137# 7280#
CCPFLGS         FBB3   139# 1032  3342  6228  7282#
CDRMAXA         2869  1823  2407  2496  2871  6728# 7474  7500  7539
CHECKA          2875  2440  6735#
CHECKALLMEDIA   1F2F  4609  4632# 6235
CHECKFCB        10CE  2251# 5215  5781  5787  6040  6046
CHECKMEDIA      1EF5  4607  4611# 4650
CHECKMEDIA1     1F08  4623# 4629
CHECKMEDIA2     1F24  4625  4628#
CHECKNPR1       1A68  3887  3894#
CHECKNPR10      1B2B  3992  3994#
CHECKNPR10A     1B42  3998  4000#
CHECKNPR11      1A72  3896  3903#
CHECKNPR1A      1A6C  3897# 3907
CHECKNPR1B      1A6F  3900# 3996
CHECKNPR2       1A88  3908  3910#
CHECKNPR21      1AA7  3921  3924#
CHECKNPR23      1AA8  3923  3926#
CHECKNPR4       1AC2  3942# 3948  3959
CHECKNPR45      1AF5  3950  3952  3955  3960#
CHECKNPR5       1AFD  3938  3963  3965#
CHECKNPR6       1B05  3968# 3970
CHECKNPR7       1B0D  3969  3971#
CHECKNPR8       1B1C  3930  3975  3981#
CHECKNPR9       1B1D  3976  3983#
CHECKNPRS       1A4D  3853  3872# 4214
```

```
CHECKRODIR       1128  2350# 3203  3205  3412  6370
CHECKROFILE      112B  2354# 4062
CHECKSUM         1188  2429# 2689
CHECKWILD        172D  3344# 3378  5489  5805  6008  6355  6501  6533
CHECKWILD0       1730  3347# 3392
CHECKWRITE       1139  2364# 2592  4043
CHEKFCB          10D8  2256  2281# 5706
CHEKFCB1         10E2  2289  2295#
CHKAM1           1F3C  4644# 4653
CHKAM2           1F51  4645  4652#
CHKBUFFERSIZE    063B  1054# 1379
CHKCOLUMN        05C0   920   938  1014# 1077  1103  1156  1164  1419
CHKEXITFXS       1213  1900  2518# 2695  4627
CHKINVFCB        12E8  2638# 2642  3542  4324
CHKMEDIA1        10F7  2312  2315#
CHKMEDIA2        10F9  2318# 2522
CHKMEDIAFCB      10E3  2278  2302#
CHKMEDIAFLAG     0EB6  1875  1880#
CHKPASSWORD      1F78  3382  3442  4695# 6358
CHKPW            1F7E  4698#
CHKPWE1          1FD2  4716  4721  4723  4725#
CHKPWE2          1FE4  4714  4719  4732#
CHKPWE3          1FFD  4734  4739#
CHKPWERROR       1F85  3225  3382  3442  4702# 5618  5856  6358
CHKSIZ           288B  1915  2433  2506  4530  6767#
CHKSIZEQ8000H    0EEB  1887  1912# 2843
CHKWILD          1739  3214  3348  3351# 7398
CHKWILD1         173B  3353# 3355
CHKXFCBPASSWORD  20D4  3224  4838# 6468  6557
CHKXFCBPASSWORD1 20D7  4840# 5854
CLEARRIGHT       061B  1046# 1048  1085  1269
CLOSE            183D  3529# 3730  4269  4345  5698  5706
CLOSE1           184B  2262  3541# 4016  5702
CLOSEFCB         186C  3557  3562#
CLPERRCDE        FBAC   131#  650  6652  6653  7274#
CLPFLGS          FBAA   130# 7273#
CLREXT           115A  2392# 4577  5744  5797
CLRMODNUM        1154  2387# 5470  5744  5798
CMPPW            2002  4700  4743# 4841  5617
CMPPW1           200F  4748# 4750  4751
CMPPW2           201D  4745  4753#
CMPPW3           2027  4756# 4758
CMPPW4           2034  4757  4760# 4761
COLUMN           FBB7   144#  891   943   970  1015  1044  1047  1051  1078  1083
                       1100  1127  1191  1220  1239  1292  1355  7288#
COMCHR           0ABF   953  1068  1143  1209  1253  1378  1396  1421  1595#
COMMONBASE       FBF9   186# 6330  7101  7114  7334#
COMPARE          0E1F  1793# 1796  2558  2569  4086  4773  4864  5015  5133  5303
                       7032  7174  7592
COMPAREEXTENTS   18F8  3606  3648#
COMPAREMODEXT    2671  6377  6418#
COMPARERR        1DFE  4444# 4469  6366
COMPCDR          1169  2404# 2416  3005  4629
COMPCOL          0ABC   871  1193  1241  1352  1356  1361  1582#
COMPEXT          1486  2914# 3051  3727  4330  6425
COMPOUT          04F6   871   883   888#
```

```
COMPUTECS       10B8  2192# 2208# 2439
COMPUTECS0      10BE  2196# 2203  2213# 2215  2226  2235
COMPUTECS1      10C1  2199# 2201
COMPUTERR       1DCB  4426# 4465  4481  6364
CONB0           047F   809   829# 1521
CONB1           049A   761   843# 1517
CONB2           049D   839   840   846#
CONB3           04A0   828   848# 1283
CONBRK          041B   776# 1535
CONBRK1         0450   778   790   808#
CONBRK2         045E   745   811   814#  828
CONBRK3         0475   823   826#
CONBRKX         042F   786#  878
CONBUFFADD      FBBA   147#  996  1009  1111  1452  7291#
CONBUFFLEN      FBBC   148# 1009  7292#
CONECH          03C2   734# 1456
CONECH0         03C6   736#  745   747   748
CONECH1         03D6   738   742#
CONECH2         03E5   744   746#
CONIN           03B7   727#  738   996  1010  1488
CONINF          FB74    69#   77#  731   804   819  1521  7649# 7650
CONINRFLG       FBBE   149# 7293#
CONLINE         FBB9   146# 7290#
CONMODE         FBCF   160#  774   824   834   855   875   929  1401  1510  1550
                      1551  7304#
CONOUT          04CD   868#  916   929   932   942   966   972   978   978  1048
                      1052  1222  1453
CONOUTF         FB7A    70#   78#  850   863   866   956   960  1062  1374  1479
                      1577  7650# 7651
CONOUTRFLG      FBC0   150# 7294#
CONOUTSTF       FF33    94#
CONPAGE         FBB8   145# 7289#
CONSTF          FB6E    68#   76#  762   795  1519  7648# 7649
CONSTX          03FF   760#  777   782  1482  1484
CONTRAN         FBCD   159# 7303#
CONWIDTH        FBB6   143# 1015  7287#
COPYALV         1377  2734# 2847  2880  6246
COPYCBUFF       05EE  1031# 1105  1450
COPYCBUFF1      060A  1033  1038#
COPYCBUFF2      060F  1036  1040#
COPYCRINIT      28BC   231  1754  1757  6895#
COPYCRONLY      FD1D   204# 1754  1757
COPYDIR         16F7  3318# 3421
COPYDIR0        16F9  3322# 3710
COPYDIR1        16FD  3325# 3329
COPYDIR2        1710  3323  3330# 3455
COPYDIRLOC      1480  2907# 3237  3430  3433  3465  6011  6375
COPYSTAMP       280E  6592  6607  6610#
COPYUSERNO      1748  3357# 3388  3403  3431
COPYXBUFF       05D8  1023# 1309  1448
CR              000D    46#  477   546   754   906   978  1150  1453  5402
CRLF            0582   594   626   967   976# 1103  1271
CRLFP           056A   963# 1313  1330  1336
CRLFP0          0572   969#  973
CSEARCH         229D  5738# 5771
CSEARCH1        22AF  5740  5742#
```

```
CSEARCH2        22BE  5743  5745#
CSEARCH3        22C3  5741  5748#
CSEARCH4        22DA  5749  5755#
CTL             005E    48#  916
CTLA            0001    28#  953  1068  1188
CTLACOLUMN      0D2D  1047  1087  1100  1607#
CTLASW          0D2A   993  1067  1092  1099  1170  1208  1230  1254  1269  1331
                      1379  1420  1604#
CTLB            0002    29# 1205
CTLBL0          07DC  1207  1210#
CTLBL1          07E1  1212# 1214
CTLBL2          07F0  1213  1215#
CTLBL3          07F8  1219# 1222
CTLC            0003    30#  823   836  1403  1517  1521
CTLCSTATRET     0A68  1515  1524#
CTLE            0005    31# 1264  1425
CTLF            0006    32# 1069  1154  1422
CTLG            0007    33# 1229  1378  1396
CTLH            0008    34#  757   900   960  1052  1222  1235
CTLHACT         FBCA   156# 1236  7300#
CTLK            000B    35# 1224
CTLOUT          0513   911# 1080  1345  1393
CTLP            0010    36#  748   840  1279
CTLQ            0011    37#  747   827   839
CTLR            0012    38# 1318
CTLRL1          08ED  1324# 1327
CTLRL2          08F8  1325  1328#
CTLS            0013    39#  744   790   809
CTLU            0015    40# 1307
CTLW            0017    41# 1070  1161  1185  1208  1330  1421
CTLWL0          0755  1164  1166#
CTLWL1          0760  1163  1169#
CTLWL13         076A  1165  1172#
CTLWL15         076F  1158  1174# 1178  1182
CTLWL2          0773  1171  1176#
CTLWL25         0782  1177  1180#
CTLWL3          0791  1168  1184#
CTLWSW          0D2B  1121  1141  1173  1177  1182  1185  1330  1435  1438  1605#
CTLX            0018    42# 1288
CTLZ            001A    43#
CURBCBA         28F0  2601  6951# 7024  7065  7130  7135  7174  7176  7205  7208
                      7210  7214  7222
CURDMA          FD1B   203# 2003  2615  6714# 7024  7089
CURDSK          28A9  1713  2332  2345  4504  4648  5011  5066  5220  5335  5347
                      5353  5359  5361  5380  5436  6086  6106  6111  6113  6125
                      6130  6135  6832# 6863# 7048
CURRECA         286D  1825  1852  1959  1990  6730#
CURSELECT       1E6B  2532  4503# 4598  4654  5741  5984  6015  7054
CURTRKA         286B  1824  1851  1957  1987  6729#
DCNT            FBE1   175# 1933  2406  2533  2629  2648  2659  2668  2668  2677
                      3044  3140  3164  3660  4621  4630  4883  5079  5751  5752
                      5754  6908# 7321# 7506  7559  7560  7602
DEBLOCK         2958  2604  6974  6999  7003#
DEBLOCK1        298D  7030  7034#
DEBLOCK10       2A61  7036  7123  7132# 7159
DEBLOCK11       2A64  7134#
```

```
DEBLOCK12       2A69  7102  7115  7139#
DEBLOCK15       299A  7037  7039#
DEBLOCK2        29C2  7028  7038  7055#
DEBLOCK25       29D6  7059  7062#
DEBLOCK3        29E0  7061  7068#
DEBLOCK35       29E3  7067  7073#
DEBLOCK4        29E6  7075#
DEBLOCK45       29F2  7032  7084#
DEBLOCK5        29FC  7087# 7088
DEBLOCK6        2A10  7091  7093#
DEBLOCK7        2A33  7099  7104#
DEBLOCK75       2A48  7114  7117#
DEBLOCK8        2A51  7016  7126#
DEBLOCK9        2A58  7026  7045  7076  7129# 7174
DEBLOCKDTA      28FF  3869  4220  6264  6970#
DEBLOCKFLUSH    2907  6975# 7000
DEBLOCKFLUSH1   2911  6980# 6995
DEBLOCKFLUSH2   293C  6982  6985  6990  6993#
DEBLOCKIO       2A71  7050  7074  7144#
DELETE          1636  3171# 5776
DELETE00        1645  3190# 3229
DELETE01        166B  3195  3221#
DELETE02        167E  3210  3214  3223  3224  3228#
DELETE1         1684  3191  3233#
DELETE10        1687  3236# 3279  4738  5892
DELETE11        168A  3217  3238# 3400
DELETE12        169B  3243  3253#
DELETE13        169D  3250  3261#
DELETEX         163C  3178# 3225
DELIM           024D   422   437   471   515   548#
DELIM1          0252   551#  556
DELIM2          025D   552   558#
DELIM3          0269   566   568#
DELIMITERS      023F   546#  550
DEVINITF        FF3F    98#
DEVTBLF         FF3C    97#
DFPASSWORD      28E7  4770  4781  6628  6943#
DIOCOMP         0EA5  1865  1872# 6237  6261
DIRBCBA         2879  2547  2597  4175  6737#
DIRBIOS1        2562  6302  6304#
DIRBIOS2        256B  6305  6307#
DIRBIOS3        258B  6303  6324#
DIRBIOS4        2590  6306  6327#
DIRBLK          2889  2865  6766#
DIRCNT          2898   227  3887  3889  3996  6783#
DIRINP          0A20  1476  1483#
DIRINP1         0A25  1478  1487#
DIRLOC          289D  2911  2969  3077  6791#
DIRMAX          2887  2495  2667  6765# 7476
DIRREC          0004  2014# 2016
DIRSTAT         0A17  1477  1480#
DIRTOUSER       12C3  2617# 5757
DISCARD         125B  2544  2549#
DISCARD0        125D  2541  2551#
DISCARD1        1265  2556# 2566
DISCARD2        1274  2559  2562#
```

```
DISCARDDATA      1252  2462  2543# 2837  6252
DISCARDDATABCB   124A  2540# 4196  4231
DISCARDDIR       1258  2546# 2838  4605  4620
DISKF            000C   314#
DISKREAD         19F5  3808# 4416
DISKREAD0        1A00  3818# 3824  3846
DISKSELECT       1E51  4486# 4508  4648  5459  6030
DISKSELECT1      1E54  4489# 7048
DISKWR1          1C11  4105  4161#
DISKWR10         1C15  4099  4166#
DISKWR11         1C84  4167  4168  4211#
DISKWR2          1CC7  4245  4261#
DISKWRITE        1B63  4039# 4423
DISKWRITE0       1B9C  4070  4073#
DISKWRITE15      1BB5  4086  4089#
DISKWRITE2       1BBA  4083  4106#
DISKWRITE3       1CDE  4278  4284#
DISKWRU          1C0F  4152  4158#
DLOG             2865  1892  2329  4492  4643  5425  5432  5961  6069  6070  6239
                       6700#
DMAAD            FBD8   168#  687   690   695  1110  2608  5451  5979  7094  7314#
DMINX            289F  2087  2942  3508  3938  4122  4148  6399  6794#
DMPOS0           0FAB  2045# 2046
DMPOS1           0FB9  2052# 2056
DMPOS2           0FC2  2055  2057#
DMPOSITION       0FA4  2041# 2086  3935  4121  6390
DMSET            18A8  3584  3595#
DOCTLA           079F  1189#
DOCTLB           07CA  1206#
DOCTLF           071C  1155#
DOCTLF0          0724  1157# 1231
DOCTLH           0839  1237# 1250  1254  1396
DOCTLU           08CE  1312# 1322
DOCTLW           073A  1162# 1209
DOESXFCBEXIST    1608  3156# 3400  3430
DONEWLINE        09C0  1420  1424#
DONTWRITE        1CBA  4216  4221  4241#
DORUBOUT         085C  1236  1251#
DPBADDR          2873  1830  1833  6016  6734# 6743
DPBLIST          0011  1835  6771#
DPTR             28B7  2373  2533  2682  6889#
DREC             2891  1935  2433  2442  6775#
DRVLBL           1474  2892  2903#
DRVLBLA          286F  1826  2861  2905  4685  4898  6474  6731#
DRVRELOG         1233  2531# 4627
DRVTBLF          FF42    99#
DSKERR           0D3D   595  1630#
DSKMAP           0010  2029# 2070  2773  2775  3358  3421  3567  3568  4146
DSKMSG           0D2F   596  1629#
DSKMSK           0003  2016# 2629  2677
DSKSHF           0002  1934  2015#
DSPLFLGS         FBA6   126# 7268#
DTABCBA          287B  2541  2544  4660  6738# 6971  7000
ECHOC            03F0   738   751#  913
EFCB             2862  3682  6698#
EMPTY            00E5  2010# 2889  3001  3013  3257  6383  6698
```

```
EMPTYBCBA     28F6  6957# 7170  7177  7199
ENDDIR        FFFF  1689# 2659
ENDOFDIR      12F8  2645# 2820  2880  2997  5541  5820  5871  7495  7534
ENDSEARCH     159B  3024  3067#
ENDSEARCH1    15AD  3071  3074#
ENTSP         0332   247   654# 1659# 6347  6693
ERRDRV        FBED   182# 1725  7328#
ERRFLG        0280   285   592#
ERRFLG1       02AF   611   613#  615
ERRFLG2       02B8   614   616#
ERRFLG3       02E5   620   625#
ERROR         FBFB   187# 1726  7335#
ERRORMODE     FBE7   179#  286  1726  6188  7325#
ERRORSUB      007C   281# 7335
ERRTBL        014A   283   336#
EXPAND        05C9  1017# 1443
EXPAND1       05CE  1019# 1021
EXTMSK        28B4  2158  2918  2955  4905  6763# 7447
EXTNUM        000C  2019# 2133  2904  2975  3029  3161  3395  3421  3454  3493
                    3523  3601  3615  4328  4356  4431  4788  4816
EXTVAL        28A6  2051  2160  6824#
FALSE         0000  1688# 1842  2996  4040  4421  4624
FCBDSK        28BF  4590  6665  6917#
FCBLEN        0020  2013# 2014  2030  2031  2775  3568  3601  3703  3710
FCBNZERO      1886  3575  3578#
FCBSHF        0005  2017# 2678
FILEEXISTS    23E3  3396  5822  5935#
FILL0         1C4B  4190# 4191  4192
FILL00        1C37  4179# 4180
FILL1         1C5B  4195# 4206
FINDXFCB      FD16   199# 3019  3154  6922#
FIXHASH       2CC8  3278  3424  3714  6383  6483  7596#
FLUSH         251A  5711  6259#
FLUSH0        24E8  6238# 6351
FLUSH1        24ED  6240# 6256
FLUSH3        2506  6243  6247#
FLUSH35       2510  6246  6251#
FLUSH4        2513  6250  6253#
FLUSH5        2515  6241  6255#
FLUSHF        FF48   101# 6236  6260
FLUSHX        2520  3999  6248  6262#
FREEMODE      289E  5141  6120  6123  6792#
FUNC1         03C2   311  1456#
FUNC10        06C8   313  1505#
FUNC100       26A6   329  6441# 6487#
FUNC101       271F   329  6492#
FUNC102       2728   329  6498#
FUNC103       2764   329  6488# 6528#
FUNC104       27F1   330  6585#
FUNC105       27FE   330  6595#
FUNC106       2813   330  6622# 6634#
FUNC107       281E   330  6638#
FUNC108       2827   331  6648#
FUNC109       0A7B   331  1547#
FUNC11        0A3D   313  1508#
FUNC110       0A89   331  1553#
```

```
FUNC111         0A95   331  1560#
FUNC112         0A95   332  1561#
FUNC12          217D   315  5413#
FUNC13          2183   315  5422#
FUNC14          2196   315  5455#
FUNC15          21A0   315  5468#
FUNC16          2287   316  5687#
FUNC17          229B   316  5735#
FUNC18          22EC   316  5762#
FUNC19          22F5   316  5773#
FUNC2           052B   311  1459#
FUNC20          22FB   317  5778#
FUNC21          2304   317  5784#
FUNC22          230D   317  5790#
FUNC23          23EF   317  5954#
FUNC24          23F5   318  5959#
FUNC25          23FB   318  5963#
FUNC26          2401   318  5967#
FUNC27          2408   318  5982#
FUNC28          11D3   319  5989# 5998#
FUNC29          2411   319  6002#
FUNC3           0A01   311  1462#
FUNC30          2417   319  6006#
FUNC31          2423   319  6013#
FUNC32          242D   320  6020#
FUNC33          2441   320  6037#
FUNC34          244A   320  6043# 6145
FUNC35          2453   320  6049#
FUNC36          1E40   321  6054#
FUNC37          2459   321  6056#
FUNC38          0A75   321  6090# 6140#
FUNC39          0A75   321  6117# 6141#
FUNC40          244A   322  6145#
FUNC42          0A75   322  6150# 6157#
FUNC43          0A75   322  6152# 6158#
FUNC44          247D   323  6162#
FUNC45          248B   323  6175#
FUNC46          2490   323  6193#
FUNC47          24CD   323  6223# 6227#
FUNC48          24DF   324  6234#
FUNC49          252A   324  6268# 6272#
FUNC49SET       2545  6276  6279#
FUNC50          2550   324  6286# 6290#
FUNC50RET       25B0  6297  6336#
FUNC6           0A07   312  1474#
FUNC7           0A2B   312  1491#
FUNC8           0A31   313  1495#
FUNC9           0A37   313  1499#
FUNC98          24EB   328  6351#
FUNC99          25C8   328  6353#
FUNCRET         0A75  1540# 1657# 4869  6140  6141  6149  6157  6158  6223  6268
                      6286  6634
FUNCTAB         00C6   257   310#  314   325
FUNCTAB2        012C   263   327#  334
FWFMSK          0080  2026# 2400  3540  4030
FX              FBDF   173#  226   609   669   682   702   863   878   926   937
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
                      1126  1569  1737  1896  2169  2509  4167  4322  4626  4727
                      4801  6243  7319# 7390  7489
FXSTSMSG        0D85   344  1644#
GETALLOCBIT     134E  2698# 2720  3294  3301  3958
GETATTS         1018  2109# 3173  3374  3438  5493  5692  5794
GETATTSLOOP     1021  2115# 2119
GETBCB1         2AB0  7172# 7193
GETBCB11        2ACE  7176  7178#
GETBCB14        2AF3  7177  7183  7185  7186  7188#
GETBCB15        2AF4  7180  7190#
GETBCB2         2B04  7192  7194#
GETBCB25        2B14  7197  7200#
GETBCB3         2B17  7199  7202# 7225
GETBCB4         2B43  7174  7212#
GETBCB5         2B59  7216  7218  7220#
GETBCBA         2A92  7021  7164#
GETBLOCK        16A8  3281# 4131
GETBUFFA        1282  2574# 2611  7024
GETDE0          149F  2932# 2955
GETDE1          14A2  2936# 2939
GETDE2          14AC  2938  2941#
GETDE3          14B9  2944  2946#
GETDIREXT       1497  2151  2926# 3482  3548  3602  3796  4332  5822  6393
GETDIRMODE      1F73  3210  3223  4684# 4696  5592  5850  5880  6496  6532
GETDM           0FCC  2072# 2088  3941  3944  4127
GETDMA          0FC4  2069# 2074  2639  6433  6504
GETDMD          0FDA  2077  2079#
GETDPTRA        1140  2352  2369# 2771  2882  3009  3195  3239  3337  3477  3637
                      4464  4706  4815  4830  4869  6363  6383  6415  7601
GETDTBA         2112  3706  4850  4878# 5910  6505
GETDTBA8        2110  3265  4716  4874# 5596  5605  6544  6572
GETEXTA         103F  2124  2131# 2157  2177  2394  2957  3559  3725  4906  4916
                      5743  6454  6521  6535  6559  6676
GETFCB          1058  2146# 3785  3817  4068  5261  6390
GETFCB0         106D  2150  2152#
GETFCB1         1074  2153  2155#
GETFCBA         104F  2140# 2148  2165  2929  3460  5542  6395
GETFCBADDS      2667  3567  6400  6414# 6419
GETFILESIZE     1E13  4455# 6052
GETHASH         2BE0  7361  7391  7393  7398  7403# 7601
GETHASH1        2BF1  7411# 7433
GETHASH3        2C06  7413  7415  7422#
GETHASH4        2C07  7419  7424#
GETHASH5        2C11  7426  7428#
GETHASH6        2C1E  7407  7434#
GETHASH7        2C47  7448# 7453
GETHASH8        2C56  7449  7454#
GETHASH9        2C5B  7441  7457#
GETMODNUM       114B  2380# 2389  2398  3539  3790  4028  4035  4914  5697
GETNALBS        13DE  2743  2760  2807# 2852  6198
GETNEXTBCBA     2B78  4180  6995  7167  7169  7192  7204  7205  7231#
GETRCNTA        1047  2136# 2142  4022  4268
GETRRA          1037  1778  1788  2126# 4458  5674  6366
GETS1           1031  2122# 2240  2247  2277  2538
GETSIZE         1E1F  4461# 4472
GETXFCB         2083  4699  4723  4736  4809# 5601  5614  5852  5887  6518  6546
```

```
                         6577
GETXFCB1          2099   4814# 4839  5903  6569
GFC               0219    446   463   501   515#
GFC1              0232    525   528#
GOBACK            283B    253  1740  2519  5941  6657#
GOBACK0           2835   1738  1739  6655#
GOERR             0DB2   1702  1706  1715# 1877  1878  1904
GOERR1            0DB8   1724# 5943
GSP1              249D   6202# 6207
GSP2              249E   6203# 6205
GSP3              24A2   6204# 6204
GSP4              24AA   6203  6206#
HASH              FB9D    120# 7262# 7363  7405  7437  7437  7514  7598
HASHL             FB9C    119# 2467  7261# 7385  7395  7400  7469  7491  7499  7538
                         7570  7583
HASHMX            FD12    196# 6899# 7478  7504
HASHTBLA          287D   2874  6739# 7366  7463  7482  7482  7503  7600  7607
HIGHEXT           28B5   2282  4011  4050  4540  4568  4571  5479  5497  5532  5551
                         5568  5580  5922  6676  6687  6854#
HIGHFXS           005D    255   259#
HLROTL            10FE   2322# 2336  5336  5994
HLROTL0           10FF   2325# 2326
HLROTR            10AB   1934  2006  2184# 2347  2709  2810  2953  4944  5351
HLROTR0           10AC   2187# 2190
HOLDDMA           0372    679#  690   693
HOME              0E80   1847# 2815  2869  2981  4622  5480
HOMEF             FF18     85# 1849  7654# 7655
INCRPDCNT         289D   5148  6120  6132  6790#
INCRRR            0DFE    669  1777# 5239  5250
INDEX             0FE0   2084# 3842  4082  5263
INDIC0            17B4   3452# 3466
INDIC1            17CE   3456  3462#
INDICATORS        17A4   3436# 6010
INFO              FBDB    170#  224   267   381   622   684  1110  1128  2070  2112
                         2128  2133  2138  2226  2294  2309  2384  2965  3214  3333
                         3346  3358  3478  3681  3682  3689  4061  4146  4297  4551
                         4589  4601  4709  4712  4726  4737  4741  4810  4813  4830
                         5344  5373  5373  5425  5535  5555  5577  5768  5971  6062
                         6106  6121  6332  6340  6416  6452  6457  6665  7316#
INITAPOS          06A8   1090# 1118  1321
INITHASH          2B8F   2885  7358#
INITIAL0          140D   2856# 2859
INITIAL2          143B   2878# 2887  2890  2901
INITIAL3          146E   2893  2899# 2905
INITIALCS         11C6   2446  2481#
INITIALIZE        13E8   2824# 4521
INITIALIZE2       1405   2843  2846  2848#
INITXFCB          20B1   4823# 5895  6462  6554
INITXFCB0         20B7   4826# 6469
INITXFCB1         20CC   4834# 4836
INITXFCBSEARCH    15FD   3151# 3384  3406
INITXFCBSEARCH1   15FF   3153# 3159  3179
INITXPOS          06B3   1095# 1117  1179
INVIS             000A   2037#
KBCHAR            FD20    206#  729   761   778   790   811   830  1517  1522  1587#
LAST              2CFD   7245# 7246  7618# 7619  7625# 7626
```

```
LASTBCBA        28F2  6955# 7166  7177  7187  7193  7201  7204  7228
LASTBCBLINKSDE  2B6D  7206  7227#
LASTBLOCK       2896  2596  3868  4143  6779#
LASTDRIVE       2895  4085  4144  6778#
LASTOFF         2894  4088  4143  4186  6777# 7061
LEFTTST         16D2  3297# 3315
LF              000A    47#  755   978  1151  5402
LINELEN         090B  1194  1203  1246  1333#
LINFO           289E   243  4593  6022  6250  6793#
LISTCP          FBD4   162#  856   886  7308#
LISTF           FB80    71#   79#  312   886  1570  7651# 7652
LISTSTF         FF2D    92# 7661# 7662
LOCKUNLOCK      289C  5224  5235  5244  6151  6153  6787#
LOGFXS          28C0  2507  6926#
LRET            0D9F  1675# 2263  2628  3078  3531  3732  3834  4019  4072  4348
                      4362  4812  4922  5709
LRETEQFF        15C0   261   262   322  1482  2525  3091# 3646  5580  5892  5938
                      6163  6164  6361  6367  6368  6454  6487  6488  6532
LSNADD          2871  1827  2277  2485  2516  2537  2846  4524  6732#
LSTFCB          001F  2030#
LSTOUTRFLG      FBC6   153# 7297#
LSTREC          007F  2011# 4069
MAKE            1916  3675# 3761  4370  5864  5889  6461  6552
MAKE0           194C  3704# 3705  3707
MAKE00          2397  5887  5889  5894#
MAKE2           23AB  5898  5900#
MAKE3A          23D2  5880  5882  5884  5910  5916#
MAKEA0          2333  5820  5823#
MAKEFCBINV      12DD  2631# 3645
MAKEFLAG        289C  5474  5678  5830  5839  5841  5921  6788#
MAKEX04         2359  5848  5850  5852  5854  5858#
MAKEXFCB        FD15   198# 3694  4583  5889  6460  6549  6921#
MAXALL          2885  1837  2798  2808  3289  3954  6210  6764#
MAXEXT          001F  2020# 2922  2958  3733
MAXMOD          0040  2024#
MD              0000  1621# 1625# 1630  1631  1632  1633  1634  1641  1645  1648
MEDIACHANGE     11B2  1893  2461#
MEDIAFLAG       FBF0   183# 4639  7330#
MERGE0          1878  3569# 3597
MERGED          1893  3570  3585#
MERGERR         18F2  3542  3583  3590  3592  3643#
MERGEZERO       181F  3511# 3587  3588
MODNUM          000E  2022# 2384  3031  4325  4436  6419
MOVE            0E29   623   624  1027  1039  1786  1802# 1831  1835  3455  3480
                      3672  4708  4833  4865  5068  5079  5327  5675  6409  6509
                      6511  6611  6646  7046  7077  7160
MOVE0           0E2A  1806# 1810
MOVEF           FF4B   102#  211   212  2625  2745  2746  6333  7368  7599  7609
                      7612
MOVEOUT         FD09   193#  211# 7366  7607
MOVETPA         FD0C   194#  212# 7101  7103  7107  7121
MPM             0000    21# 1636  1651  1668  1719  1731  1742  1855  1923  1937
                      2206  2253  2284  2297  2450  2464  2475  2619  2812  2829
                      2989  3040  3103  3109  3142  3197  3244  3373  3414  3448
                      3533  3544  3632  3662  3668  3751  3765  3775  3781  3812
                      3823  3844  4008  4052  4075  4091  4108  4138  4249  4266
```

```
                         4352  4374  4382  4510  4529  4560  4764  4775  4854  4926
                         5416  5424  5442  5457  5472  5491  5508  5520  5526  5550
                         5573  5583  5627  5691  5713  5801  5808  5814  5826  5866
                         5873  5919  5940  5946  5969  5987  6028  6059  6072  6081
                         6088  6147  6167  6177  6221  6266  6284  6587  6599  6613
                         6624  6640  6661  6672  6702  6718  6796  6816  6830  6847
                         6861  6904  6919  6930  6945  7241  7614
MRGRC1           18D9    3620  3624  3628#
MRGRC2           18DA    3622  3625  3630#
MULTCNT          FBE6     178#  236   665   699  4096  5205  5227  5232  5241  5272
                         6099  6165  6849# 7324#
MULTFXS          0095     237   289#
MULTIO           0348     666#  671
MULTIOF          FF45     100# 4005
MULTNUM          2899     667  3907  3991  6784#
NAMLEN           000F    2027# 2973  3168  3703  5536  5747
NEWCHECKSUM      1186    2427# 2593
NEWCTLACOL       0D2E    1083  1087  1608#
NFUNCS           0033     255   325#
NFUNCS2          000F     262   334#
NOPBLOCK         1BD1    4124  4129#
NORMALSTATUS     0A6F    1510  1533#
NOSELECT         1EDF    4594  4597#
NOSELECT0        1EE9    4602# 5741
NOTBACKSP        050D     900   904#
NOTC             09A0    1401  1405  1409#
NOTCTLA          07C5    1188  1204#
NOTCTLB          080B    1205  1223#
NOTCTLF          0735    1154  1160#
NOTCTLG          082D    1229  1232#
NOTCTLK          081E    1224  1228#
NOTCTLW          079A    1161  1187#
NOTE             0892    1264  1277#
NOTH             0850    1235  1247#
NOTP             08A2    1279  1286#
NOTR             094D    1159  1186  1318  1367#
NOTRUB           0873    1249  1262#
NOTU             08D5    1307  1316#
NOTX             08C4    1288  1304#
NOUPDATE         1CCF    4263  4267  4273#
NOWRITE          111A    2340# 2366  2459  3537  4724  4734  4898  5609
NXTLINE          0708    1142# 1425
NXTREC           0020    2031# 2032  2143  3479  4316  4480
OFF              0000      20#   21
OFFSET           288D    1983  6768#
OLDDSK           FBDA     169#  228  5440  5465  7315#
OLOG             FB90     112# 2502  5685  7248#
ON               FFFF      19#   22
OPEN             17DA    3468# 4361  5513  5612
OPEN1            17DD    3471# 5537
OPENCOPY         17DE    3474# 3773
OPENMOD          1999    3738#
OPENREEL         1971    3717# 3832  4071
OPENREEL0        19A0    3736  3748#
OPENREEL1        19B9    3756  3771#
OPENREEL2        19BC    3770  3779# 3805
```

```
OPENREEL3       19DA  3728  3794#
OPENREEL4       19EC  3798  3803#
OPENRERR        19C9  3745  3759  3763  3788#
OPENUSERZERO    21D3  5481  5533#
OPENX           21E7  5513  5538  5540#
OPENX0          2212  5553  5555  5557  5559  5568  5578  5581#
OPENX0A         224E  5594  5596  5611#
OPENX0B         2257  5601  5615#
OPENX1          2272  5614  5617  5619  5622# 5927
OPENX1A         2276  5592  5598  5605  5610  5624#
OPENX2          227B  5683# 5920  5930
OPENXA          21F7  5542  5544#
OUTDELIM        FBD3   161#  983  1554  7307#
PAD             026C   398   400   402   404   573#  576
PAGEMODE        FBC8   154# 7298#
PARSE0          0170   392#
PARSE6          01B6   441#  447
PARSE8          01D4   460#  464
PARSEDRV        0198   421#
PARSENAME       01AC   417   435#
PARSEOK         01E0   423   438   448   466#  511
PARSEPW         01FB   445   462   493#
PARSEPW1        0205   500#  502
PARSETYPE       01CA   443   452#
PASSMSG         0D76   343  1638#
PCTLH           0565   950   958#
PERMSG          0D41   337  1631#
PERROR          01F5   484#  519   522
PERROR1         01F6   425   427   486#
PHYMSK          2890  3886  4174  6263  6770# 7127  7181  7237
PHYOFF          28EF  6950# 7017  7086  7216  7236
PHYSHF          288F  2006  6769#
PMDEFAULT       FBC9   155# 7299#
PRFCB           030E   619   637#
PRFCB1          0316   622   638#
PRFX            02FB   627   635#
PRFX1           030B   610   636#
PRINT           058C   601   642   981#  988  1503
PUNCHF          FF12    83#  312  7652# 7653
PWERROR         1FDF  4729# 5620  5857  6468  6557
PWFCB           28D7  4707  6937#
PWMODE          28E4  4717  4724  4733  5619  5857  5901  5912  6569  6573  6939#
QCONIN          059C   990# 1141  1145
QCONIN1         05B6  1006  1008#
QCONINX         FD21   207# 1000
QCONOUTF        04BD   861#  880
QDIRFCB1        2140  4895  4901# 5594  5848  5884
QFLAG           FBD5   163#  768  1513  7309#
QSTAMP          2132  4893# 5684  5931
QSTAMP1         2136  4896# 4912
RANCLOSE        1D5B  4324  4326  4330  4343#
RANDISKREAD     1DB9  4412# 6041
RANDISKWRITE    1DC2  4419# 6047
RANREC          0021  2032# 2128  4298  4482
RCOUNT          28A5  2156  2173  3819  3923  4022  4244  4268  6823#
RDBUFF          0E93  1861# 3864  7157
```

```
RDDIR           1292  2584# 2686  2696  5753
RDDIRFLAG       FD14   197# 7486
RDECHO          095E  1371  1375#
RDECH05         096D  1378  1380#
RDECH1          0971  1261  1386#
RDECH2          098C  1397  1399#
RDIR            130E  2662  2664# 4624
RDIR1           1340  2662  2693#
RDIR2           133D  2691# 7487
READ            06D7  1109  1114# 1296  1299  1315
READDEBLOCK     1A3F  3855  3867#
READDIR         1308  2661# 2818  2879  2996
READDIR0        131F  2674#
READDIR1        1326  2679# 2680
READDIR2        1330  2684# 2692  7568
READEN          09E7  1150  1151  1440#
READERF         FF15    84# 1464  7653# 7654
READF           FF27    90# 1864  7659# 7660
READFSW         28B4  1863  1870  1902  6843#
READI           06C8  1102# 1108# 1505
READNO          0701  1138# 1221  1276  1331  1353  1366  1374  1419  1435
READNX          06FF  1064  1135# 1156  1175  1192  1227  1230  1240  1257  1285
                      1414  1438
READX           06DD  1119# 1295
READX1          06E4  1122#
REBOOTE         0323   287   645#
REBOOTX         0329   647#  824  1407
REBOOTX0        032C   646   649#
REBOOTX1        032F   311   651# 6229  6231
RECCNT          000F  2028# 2138  2143  3493  3523  3615  4464  6363
RECORDOK        1A19  3821  3835#
RECORDOK1       1A22  3845  3850#
RECSIZ          0080  2012# 2014  2210  2624
REFRESH         065E  1066# 1120  1226  1273  1364  1398
REFRESH0        066F  1071# 1165
REFRESH05       068A  1077  1079#
REFRESH1        0677  1075# 1081
REFRESH2        0693  1076  1078  1082#
REFRESH3        0699  1084#
RELOG           FBDE   172# 1908  2467  2529  2688  2694  4625  7318#
RENAME          1752  3361# 5957
RENAME0         1788  3419# 3426  3431
RENAME1         1798  3427#
REP0            0913  1339# 1347
REP1            092E  1340  1348#
RESBDOS         FD00   192#
RESBDOSPG       FD00    56#  192   193   194   195   196   197   198   199   200
                       202   203   204   205   206   207
RESEL           FBDD   171#  245   620  4588  6659  7317#
RESELECT        1E9A  4548# 5215  5473  5689  5780  5786  6039  6045  6051
RESELECT1       1EB9  4546  4578#
RESELECTX       1E90  4539# 5486  5746  5775  5799  5956  6009  6354  6451  6500
                      6530
RESET37X        245A  2473  5438  6067#
RESETCOPYCRONLY 0DDC  1756# 3722  4113  4282  4391  4405
RESETDMA        0390   692#  703
```

```
RESETDMA1       0393   687   694#
RESETFWF        1B50  4027# 4264  4281  4287
RESETRELOG      0EE6  1898  1906#
RESETRR         0E19   702  1790# 5241  5254
RESTOREDIRFCB   162B  3166# 5603  5623  5908
RESTORERC       182D  3519# 3554  3804  4340
RESTORERC1      183B  3524  3526#
RETBLOCK        16E6  3295  3302  3306#
RETBLOCK0       16EE  3291  3312#
RETMON          2858  6659  6692#
RETSELECT       0E7E  1840  1843#
REVERSE         062B  1050# 1052  1086  1433
RIGHTTST        16BA  3288# 3298  3305
RLOG            FB92   113# 2314  2503  4513  4531  5360  5434  6082  6082  6704#
                      7249#
RMF             289C  3758  3801  3810  3921  3952  3975  3998  4040  6789#
RODERROR        0DA1  1700# 2367
RODMSG          0D67   338  1634#
RODSK           2863  2342  2493  5361  5432  6004  6070  6079  6699#
ROFERROR        0DA6  1704# 2358
ROFILE          0009  2036# 2361
ROFMSG          0D58   339  1633#
ROOTBCBA        28F4  6956# 7165  7207  7210  7222
ROTEST          1132  2356  2360# 5557
ROTL            1362  2715# 2715
ROTR            1370  2726# 2730  3309
RSEEK           1CE4  4292# 4415  4422  5259  6361
RSEEK2          1D50  4333  4334  4336#
RSEEK3          1D6B  4322  4349#
RTNPHYERRS      0DC5  1720  1729#
RUBOUT          007F    44#  892  1249  1253  1397
RUBOUTACT       FBCB   157# 1250  7301#
RWFXS           28CD  2522  6927#
RXFCB2          2753  6505  6514#
RXFCB3          275F  6509  6520#
SAVECOL         0614  1043# 1073  1391
SAVECOLUMN      0D2C  1044  1051  1606#
SAVEDCNTPOS     15F5  3133  3139#
SAVEDCNTPOS0    15EA  3072  3132#
SAVEDCNTPOS1    15ED  3013  3087  3134#
SAVEDCNTPOS2    15DE  3054  3123#
SAVEDCNTPOSRET  15FB  3138  3146#
SAVEDMA         0389   664   689#
SAVEHASH        28B8  6891# 7598
SAVEMOD         28A7  3735  3790  6825#
SAVEPOS         0ABE  1129  1207  1213  1266  1319  1345  1591#
SAVERANR        28AE  1788  6840#
SAVERR          0E09   664  1783# 5232
SAVERR1         0E0D  1785# 1791
SAVERR2         0E12  1784  1787# 1791
SAVEXFCB        28A2  3016  3020  3071  6809#
SCANDM0         13B0  2776# 2805
SCANDM1         13C3  2782  2788#
SCANDM2         13C9  2787  2794#
SCANDM3         13D8  2797  2802#
SCANDMA         13A6  2750  2763  2766# 2898
```

```
SCANDMAB        138B  2748# 3273  3275  6379  6411
SCANDMB         1390  2753# 3640
SCB             FB9C   115# 6275  7257# 7313
SCBADD          FBD6   167# 7313#
SCBPG           FB00    57#   67    68    69    70    71   112   113   115   119
                       120   121   125   126   130   131   135   136   137   138
                       139   143   144   145   146   147   148   149   150   151
                       152   153   154   155   156   157   158   159   160   161
                       162   163   167   168   169   170   171   172   173   174
                       175   177   178   179   180   181   182   183   184   185
                       186   187   188
SCFXS           28D4  2525  6928#
SCONOUTF        FB7A   865  1577#
SDL0            26BF  6453  6455#
SDL1            26E4  6458  6464#
SDL2            2703  6475# 6567
SDL3            2719  5906  6477  6482# 6562
SEARCH          14F7  2973  2976# 5759  6453  6458
SEARCH1         14FA  2979# 3395
SEARCHA         28A0  2965  2999  3160  3389  3395  5766  5768  6803# 7322#
SEARCHCHAIN     FBE8   180# 7326#
SEARCHEXT       157C  3030  3035#
SEARCHEXTNUM    14F5  2974# 3188  3234  3410  3445  4460  4812  6373  6550
SEARCHFIN       15BA  2995  2997  3007  3084#
SEARCHH0        2C87  7475  7477#
SEARCHH1        2CAE  7484  7493# 7512# 7530
SEARCHHASH      2C6B  2995  7465#
SEARCHI         14DD  2963# 2978  3168  3685  5536  5760
SEARCHI1        14E3  2966# 3161  3395
SEARCHL         FBE5   177# 2967  3010  5760  6798# 7323# 7471
SEARCHLOOP      1557  3017  3018  3023# 3066
SEARCHMOD       15B7  3031  3082#
SEARCHN         1503  2984# 3019  3062  3073  3098  3099  3100  3127  3161  3168
                      3229  3279  3425  3464  3685  4470  5536  5760  6387
SEARCHNAMLEN    14F0  2972# 3470  3564  3755  4714  4919  5891  6503  6539  6571
SEARCHNEXT      152A  3002  3008#
SEARCHNM        15C7  3033  3097#
SEARCHOK        1594  3020  3025  3027  3034  3063# 3107
SEARCHUSER0     28BD  3057  3100  4587  5506  5514  6913#
SECTOR          28FD  2001  6964# 6992  6999
SECTPT          2880  1834  1966  1970  1979  6760# 6771
SECTRAN         FF30    93# 1999  7662#
SEEK            0F0C  1954# 3858  4197  4224  7148
SEEK0           0F22  1962# 1968
SEEK1           0F39  1965  1969# 1976
SEEK2           0F50  1973  1977#
SEEKCOPY        129A  2589# 3326  3463  3641  4724  4851  5609  5912  6484  6573
SEEKDIR         0EF5  1920# 2586
SEEKERR         1DB0  4313  4348  4402#
SEEKOK          1D9C  4362  4379#
SEEKOK1         1DA1  4342  4388#
SEEKOK2         1DA4  4329  4394#
SELDSK          28A4   228  1898  2493  4501  4504  4596  5461  5465  5965  5993
                      6818# 6822#
SELDSKF         FF1B    86# 1818  7655# 7656
SELECT          1E78  4504  4507#
```

```
SELECTDISK      0E33  1812# 4494
SELERROR        0DAB  1708# 4495
SELMEMF         FF51   104#
SELMSG          0D4A   340  1632#
SEQBCBA         28F8  6958# 7170  7186  7187  7197
SEQDISKREAD     19F5  3807# 5782
SEQDISKWRITE    1B63  4038# 5788
SERIAL          0000   217# 6643
SETALLOCBIT     1368  2717# 2800
SETARECORD      0F03  1949# 7052
SETARET         23E5  3349  4047  4059  4730  5636  5642  5663  5700  5937#
SETBCBSEQ       2B83  7081  7219  7235#
SETBNKF         FF54   105# 3861  4200  4227  6325  7151
SETCDISK        1108  2330# 2503  4531  5370  5685
SETCDISK1       110B  2333# 2493
SETCDR          1176  2414# 2900  3708  4824
SETCOPYCRONLY   0DD5  1753# 3839  4285
SETCTLACOLUMN   06BA  1098# 1200  1218
SETCTLSMODE     040C   766#  792   802   817
SETDATA         12B6  2606# 3857  3999  4210  4223  5452  5980
SETDCNTDBLK     190A  3158  3167  3656# 3679  5524
SETDCNTDBLK1    190D  3658# 7501  7540
SETDIR1         12BC  2610# 4187
SETDLOG         1105  2328# 2473  4527
SETDMA          12BF   695  2608  2613#
SETDMAF         FF24    89# 2003  7658# 7659
SETENDDIR       1301  2657# 2672  2816  2873  2980  3090  4622  5480  7491  7570
SETFCB  .       1082  2163# 3854  3865  3870  4283  4289
SETFILEWF       1B57  4034# 4275  5932
SETFWF          1162  2397# 2633  3475  3639  3715
SETHASH         2BA6  2968  7376#
SETHASH1        2BB4  7382  7384#
SETHASH15       2BCF  7392  7394#
SETHASH2        2BDC  7380  7399#
SETLRET1        0A76  1543# 1653# 3792  3801  3831  3834  3849  4335
SETLSN          1241  2536# 3782  4385  5586  5625  5705  5874  5913
SETMATCH        0237   527   534#  538
SETPW           2042  4785# 5906  6480
SETPW0          204E  4790# 6631
SETPW1          2050  4792# 4798
SETPW2          205E  4793  4794  4796#
SETPW3          2076  4799  4801  4803#
SETPW4          2079  4805# 4806
SETRANDOM       1E40  4478# 6054
SETRC           17FD  2151  3491# 3556  3560  3805  4341
SETRC1          1810  3497  3501#
SETRC2          1812  3495  3503#
SETRC3          1815  3506# 3626  6397  6399
SETRLOG         11E8  2468  2500# 4525
SETRO           11D3  2491# 5995  5998
SETRR           1E0D  4452# 4469
SETSECF         FF21    88# 2002  7657# 7658
SETSIZE         1E3E  4462  4473#
SETSUBMITFLAG   1725  3335  3341#
SETTRKF         FF1E    87# 1985  7656# 7657
SETUSRCODE      243B  6022  6025#
```

```
SETXDCNT         1903  3653# 3749  4350  5806  6458  6542
SHELL            0334   240   656#
SHELLERR         0399   668   697#
SHELLRTN         03A2   663   701#
SHELLSP          0366   657   674#  704
SHELLSTK         0372   660   678#
SHL3BV           0DF6  1771# 1928  2097  6215
SHL3BV1          0DF7  1773# 1775
SHRPHYSHF        0F9D  1995  2005# 4004
SINGLE           28A3  1839  2076  2782  2943  3570  4147  6429  6814#
SKPS             0273   408   469   581#  584   586
SRCHHASH         FD0F   195# 7482
STAMP            FBF4   185# 4859  6590  6603  7333#
STAMP1           20DD  4845# 5684  6463
STAMP2           20E2  4847# 4921  5932  6466
STAMP3           20E4  4846  4849#
STAMP4           20ED  4852# 4870
STAMP5           2105  4868# 6463  6466
STARET           0A72   735  1465  1482  1489  1493  1497  1527  1530  1537# 1545
                       1557  1655# 2271  2319  2643  2911  3095  3786  4019  4135
                       4396  4410  5212  5229  5252  5419  5965  6024  6345  6496
                       6608
STHLRET          2429   379  1550  5417  5961  5985  6004  6017# 6278  6652
STRTCOL          0ABD   970  1127  1191  1221  1239  1275  1292  1583#
SUBDH            117F  2312  2422# 2435  2670  6330  6990  7224  7506  7566
TAB              0009    45#  546   585   756   932
TAB0             053D   934#  944
TAB1             0549   937   939#
TABOUT           0528   632   740   914   924#  987  1459  1572
TABOUT1          0537   926   930#
TBUFF            0080  1696# 5451
TEMPDRIVE        FBEC   181# 7327#
TEST152          015C   260   347#
TESTCTLSMODE     0413   743   772#  777   788
TESTFFFF         12FB  2639  2649# 3679  5145  5146  5723  6997
TESTHASH         2C64  7388  7462# 7467  7597
TESTMEDIAFLAG    120C  2515# 2836  4614
TESTP1           1F5F  4664# 4679
TESTPENDING      1F58  4650  4656#
TESTVECTOR       111D  1892  2314  2344# 2502  2830  4492  4513
TESTVECTOR1      1120  2346# 5360  5361
TFCB             005C  1695#
TIMEF            FF4E   103# 4863  6593  6602
TMPSELECT        1E67  4500# 5458  5464  6196  6242  6495
TOGGLEL1         04BB   855   858#
TOGGLELISTCP     04A9   849   853#
TRACK            28FB  1984  6963# 6979  6990  6992  6997  7160
TRANV            289A  1829  1998  6786#
TRUE             00FF  1687# 1839  2879  3810  4414  6361
TRUNC1           25FE  6374# 6388
TRUNC15          2619  6384# 6412
TRUNC2           261C  6377  6386#
TRUNC3           2622  6381  6389#
TSTINVFCB        12EE  2641# 3809  4064
TSTLOG0          11FE  2508# 2522  2525
TSTLOG1          1202  2510# 2512
```

```
TSTLOGFXS       11F5  2505# 2981  4605  7497  7536
TSTRELOG        122B  2528# 2696
TYPEAHEAD       FBCC   158#  782  7302#
UBYTES          000D  2021# 3027  4754
UPDATECS        11D1  2448  2487#
UPDATESTAMP     2153  4066  4910# 6372
USER0PASS       28BE  2990  2992  3054  3101  6915#
USERINFO        FD1E   205# 2307
USRCODE         FBE0   174# 1670# 4600  5505  6024  6026  7320#
UTILFLGS        FBA2   125# 7267#
VERSION         FBA1   121# 5419  7263#
VRECORD         28A8  2044  2104  2149  2167  3786  3819  3825  3905  4069  4213
                      6828#
WBOOTF          FB68    67#   75#  652  7647# 7648
WILDMSG         0D91   345  1647#
WORD            0002  1691# 6714  6720  6728  6729  6730  6731  6732  6734  6735
                      6736  6737  6738  6740  6760  6764  6765  6766  6767  6768
                      6775  6779  6786  6803  6837  6841  6865  6867  6868  6869
                      6870  6871  6872  6875  6876  6879  6899  6900  6906  6907
                      6908  6909  6951  6955  6956  6957  6958  6963  6964
WRBUFF          0E9E  1867# 4203  4239  7156
WRDIR           129A  2590# 3270  6371  6385
WRDIR0          12A2  2587  2595#
WRITE           1C9B  4217  4222#
WRITE0          1CB7  4236  4238#
WRITEF          FF2A    91# 1871  7660# 7661
WXFCB0          2791  6547# 6579
WXFCB1          27A1  6546  6555# 6581
WXFCB2          27BE  6560  6561  6563#
WXFCB3          27C6  6564  6566#
WXFCB4          27CA  6562  6568#
WXFCB5          27E4  6544  6574#
XBUFF           0BC2  1026  1033  1096  1599#
XDCNT           FD17   200# 3072  3138  3140  3154  3157  3164  3654  3657  3671
                      3679  4585  4704  5311  5506  5515  6900# 6906# 7544  7579
XDCNTEQDCNT     1624  3163# 5600  5612  5886
XDMAAD          FD19   202# 2623  4755  4789  4820  4821  5218  5897  6218  6713#
XFCBREADONLY    28B6  4046  4543  4555  5621  6669  6858#
XMOVEF          FF57   106# 7140
XPOS            0D26  1028  1096  1181  1183  1602#
ZERO            10A4  2180# 2182  6504  6518  6523
ZERODM          268B  6391  6428#
ZERODM1         2694  6429  6431#
ZERODM2         269E  6435# 6437
ZEROEXTMOD      109C  2176# 6502  6537  6570
ZPRINT          02EE   599   607   608   627   629#  633
```

```
                        title   'CP/M 3 Banked BDOS Resident Module, Dec 1982'
               ;*****************************************************************
               ;*****************************************************************
               ;**                                                             **
               ;**  B a s i c    D i s k   O p e r a t i n g   S y s t e m   **
               ;**                                                             **
               ;**  R e s i d e n t   M o d u l e  -  B a n k e d  B D O S   **
               ;**                                                             **
               ;*****************************************************************
               ;*****************************************************************

               ;/*
               ;  Copyright (C) 1978,1979,1980,1981,1982
               ;  Digital Research
               ;  P.O. Box 579
               ;  Pacific Grove, CA 93950
               ;
               ;  December, 1982
               ;
               ;*/
               ;
 001E =        ssize           equ     30
 000C =        diskfx          equ     12
 0002 =        conoutfxx       equ     2
 0009 =        printfx                 equ     9
 000B =        constatfx       equ     11
 001A =        setdmafx        equ     26
 002F =        chainfx                 equ     47
 0003 =        ioloc           equ     3

 0000                          org     0000h
 0000 =        base            equ     $

 FC00 =        bnkbdos$pg      equ     base+0fc00h
 FD00 =        resbdos$pg      equ     base+0fd00h
 FE00 =        scb$pg          equ     base+0fe00h
 FF00 =        bios$pg                 equ     base+0ff00h

 FC06 =        bnkbdos                 equ     bnkbdos$pg+6
 FC7C =        error$jmp       equ     bnkbdos$pg+7ch

 FF00 =        bios            equ     bios$pg
 FF00 =        bootf           equ     bios$pg         ; 00. cold boot function
 FF03 =        wbootf          equ     bios$pg+3       ; 01. warm boot function
 FF06 =        constf          equ     bios$pg+6       ; 02. console status function
 FF09 =        coninf          equ     bios$pg+9       ; 03. console input function
 FF0C =        conoutf                 equ     bios$pg+12      ; 04. console output function
 FF0F =        listf           equ     bios$pg+15      ; 05. list output function
 FF12 =        punchf          equ     bios$pg+18      ; 06. punch output function
 FF15 =        readerf                 equ     bios$pg+21      ; 07. reader input function
 FF18 =        homef           equ     bios$pg+24      ; 08. disk home function
 FF1B =        seldskf                 equ     bios$pg+27      ; 09. select disk function
 FF1E =        settrkf                 equ     bios$pg+30      ; 10. set track function
 FF21 =        setsecf                 equ     bios$pg+33      ; 11. set sector function
 FF24 =        setdmaf                 equ     bios$pg+36      ; 12. set dma function
 FF27 =        readf           equ     bios$pg+39      ; 13. read disk function
```

```
FF2A =          writef          equ     bios$pg+42      ; 14. write disk function
FF2D =          liststf                 equ     bios$pg+45      ; 15. list status function
FF30 =          sectran                 equ     bios$pg+48      ; 16. sector translate
FF33 =          conoutstf       equ     bios$pg+51      ; 17. console output status function
FF36 =          auxinstf        equ     bios$pg+54      ; 18. aux input status function
FF39 =          auxoutstf       equ     bios$pg+57      ; 19. aux output status function
FF3C =          devtblf                 equ     bios$pg+60      ; 20. return device table address fx
FF3F =          devinitf        equ     bios$pg+63      ; 21. initialize device function
FF42 =          drvtblf                 equ     bios$pg+66      ; 22. return drive table address
FF45 =          multiof                 equ     bios$pg+69      ; 23. multiple i/o function
FF48 =          flushf          equ     bios$pg+72      ; 24. flush function
FF4B =          movef           equ     bios$pg+75      ; 25. memory move function
FF4E =          timef           equ     bios$pg+78      ; 26. get/set system time function
FF51 =          selmemf                 equ     bios$pg+81      ; 27. select memory function
FF54 =          setbnkf                 equ     bios$pg+84      ; 28. set dma bank function
FF4E =          xmovef          equ     bios$pg+78      ; 29. extended move function

FF0C =          sconoutf        equ     conoutf         ; 31. escape sequence decoded conout
FFFF =          screenf                 equ     0ffffh          ; 32. screen function

0000 3635343332serial:          db      '654321'

0006 C32400             jmp     bdos
0009 C3E402             jmp     move$out        ;A = bank #
                                                ;HL = dest, DE = srce
000C C3F202             jmp     move$tpa        ;A = bank #
                                                ;HL = dest, DE = srce
000F C3FA02             jmp     search$hash     ;A = bank #
                                                ;HL = hash table address


                        ; on return, Z flag set for eligible DCNTs
                        ;               Z flag reset implies unsuccessful search

                        ; Additional variables referenced directly by bnkbdos

0012 0000       hashmx:                 dw      0       ;max hash search dcnt
0014 00         rd$dir:                 db      0       ;read directory flag
0015 00         make$xfcb:      db      0       ;Make XFCB flag
0016 00         find$xfcb:      db      0       ;Search XFCB flag
0017 0000       xdcnt:          dw      0       ;current xdcnt

0019 4E04       xdmaadd:        dw      common$dma
001B 0000       curdma:                 dw      0
001D 00         copy$cr$only:   db      0
001E 0000       user$info:      dw      0
0020 00         kbchar:                 db      0
0021 C37701                     jmp     qconinx

                bdos:   ;arrive here from user programs
0024 79                 mov a,c ; c = BDOS function #

                        ;switch to local stack

0025 2100002228         lxi h,0! shld aret
002B 39226605           dad sp! shld entsp ; save stack pointer
002F 31660521CB         lxi sp,lstack! lxi h,goback! push h
```

```

0036 FE0CD29401         cpi diskfx! jnc disk$func

003B 2186000600         lxi h,functab! mvi b,0
0040 09097E             dad b! dad b! mov a,m
0043 23666FE9           inx h! mov h,m! mov l,a! pchl

0047 434F505952         db      'COPYRIGHT (C) 1982,'
005A 2044494749         db      ' DIGITAL RESEARCH '
006C 3135313238         db      '151282'
0072 0000000000         dw      0,0,0,0,0,0,0,0,0,0

                   functab:
0086 03FFD502D5         dw      wbootf, bank$bdos, bank$bdos, func3
008E A400A800AC         dw      func4, func5, func6, func7
0096 E900EF0001         dw      func8, func9, func10, bank$bdos

                   func3:
009E CD15FFC3C8         call readerf! jmp sta$ret

                   func4:
00A4 4BC312FF           mov c,e! jmp punchf

                   func5:
00A8 4BC30FFF           mov c,e! jmp listf

                   func6:
00AC 7B3CCAC300         mov a,e! inr a! jz dirinp       ;0ffh -> cond. input
00B1 3CCABD00           inr a! jz dirstat               ;0feh -> status
00B5 3CCAC800           inr a! jz dirinp1               ;0fdh -> input
00B9 4BC30CFF           mov c,e! jmp conoutf            ;        output
                   dirstat:
00BD CDCE00C3C8         call constx! jmp sta$ret
                   dirinp:
00C3 CDCE00B7C8         call constx! ora a! rz
                   dirinp1:
00C8 CDD800C3C8         call conin! jmp sta$ret

                   constx:
00CE 3A2000B73E         lda kbchar! ora a! mvi a,0ffh! rnz
00D5 C306FF             jmp constf

                   conin:
00D8 2120007E36         lxi h,kbchar! mov a,m! mvi m,0! ora a! rnz
00E0 C309FF             jmp coninf

                   func7:
00E3 CD36FFC3C8         call auxinstf! jmp sta$ret

                   func8:
00E9 CD39FFC3C8         call auxoutstf! jmp sta$ret

                   func9:
00EF 424B               mov b,d! mov c,e
                   print:
00F1 21D305             lxi h,outdelim
```

CP/M-PLUS V3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
00F4 0ABEC8             ldax b! cmp m! rz
00F7 03C54F             inx b! push b! mov c,a
00FA CD5301             call blk$out0
00FD C1C3F100           pop b! jmp print

                  func10:
0101 EB                 xchg
0102 7DB4C21001         mov a,l! ora h! jnz func10a
0107 212C0422BA         lxi h,buffer+2! shld conbuffadd
010D 2ADB05             lhld dmaad
                  func10a:
0110 E5112A04D5         push h! lxi d,buffer! push d
0115 06004E0303         mvi b,0! mov c,m! inx b! inx b! inx b
011B EBCD4BFF36         xchg! call movef! mvi m,0
0121 D1D50E0A           pop d! push d! mvi c,10
0125 CDD502             call bank$bdos
0128 3A2B044F06         lda buffer+1! mov c,a! mvi b,0
012E 0303               inx b! inx b
0130 D1E1C34BFF         pop d! pop h! jmp movef

                  func111:
                  func112:
0135 322404             sta res$fx
0138 EB5E235623         xchg! mov e,m! inx h! mov d,m! inx h
013D 4E2346EB           mov c,m! inx h! mov b,m! xchg
                        ; hl = addr of string
                        ; bc = length of string
                  blk$out:
0141 78B1C8             mov a,b! ora c! rz
0144 C5E54E             push b! push h! mov c,m
0147 117001D5           lxi d,blk$out2! push d
014B 3A2404FE70         lda res$fx! cpi 112! jz listf

                  blk$out0:
0153 3ACF0547E6         lda conmode! mov b,a! ani 2! jz blk$out1
015C 78E614CA6A         mov a,b! ani 14h! jz blk$out1
0162 E610C20CFF         ani 10h! jnz sconoutf
0167 C30CFF             jmp conoutf

                  blk$out1:
016A 590E02C3D5         mov e,c! mvi c,conoutfxx! jmp bank$bdos

                  blk$out2:
0170 E123C10B           pop h! inx h! pop b! dcx b
0174 C34101             jmp blk$out

                  qconinx:
                        ; switch to bank 1
0177 3E01CD51FF         mvi a,1! call selmemf
                        ; get character
017C 46                 mov b,m
                        ; return to bank zero
017D AFCD51FF           xra a! call selmemf
                        ; return with character in A
0181 78C9               mov a,b! ret

```

```
                      switch1:
0183 118D01D5                   lxi d,switch0! push d
0187 3E01CD51FF                 mvi a,1! call selmemf! pchl
                      switch0:
018D 47AFCD51FF                 mov b,a! xra a! call selmemf
0192 78C9                       mov a,b! ret

                      disk$func:
0194 FE33DAB001                 cpi ndf! jc OKdf ;func < ndf
0199 FE62DABC02                 cpi 98! jc badfunc ;ndf < func < 98
019E FE71D2BC02                 cpi nxdf! jnc badfunc ;func >= nxdf
01A3 FE6FCA3501                 cpi 111! jz func111
01A8 FE70CA3501                 cpi 112! jz func112
01AD C3CF01                     jmp disk$function

                            OKdf:
01B0 FE11CAC701                 cpi 17! jz search
01B5 FE12CACB01                 cpi 18! jz searchn
01BA FE1AC2CF01                 cpi setdmafx! jnz disk$function

                                ; Set dma addr
01BF EB22D80522                 xchg! shld dmaad! shld curdma! ret

                            search:
01C7 EB22E305                   xchg! shld searcha

                            searchn:
01CB 2AE305EB                   lhld searcha! xchg

                      disk$function:

                      ;
                      ;         Perform the required buffer tranfers from
                      ;         the user bank to common memory
                      ;

01CF 21E203                     lxi h,dfctbl-12
01D2 79FE62DADB                 mov a,c! cpi 98! jc normalCPM
01D8 21B303                     lxi h,xdfctbl-98
                            normalCPM:
01DB 0600097E                   mvi b,0! dad b! mov a,m

                      ; ****  SAVE DFTBL ITEM, INFO, & FUNCTION *****

01DF 47C5D5                     mov b,a! push b! push d

01E2 1FDAED01                   rar! jc cpycdmain                   ;cdmain test
01E6 1FDAFC01                   rar! jc cpyfcbin                    ;fcbin test
01EA C31302                     jmp nocpyin

                            cpycdmain:
01ED 2AD805EB                   lhld dmaad! xchg
01F1 214E040110                 lxi h,common$dma! lxi b,16
01F7 CD48FF                     call movef
01FA D1D5                       pop d! push d
```

```
                        cpyfcbin:
01FC AF321D00              xra a! sta copy$cr$only
0200 212A040124            lxi h,commonfcb! lxi b,36
0206 CD4BFF                call movef
0209 112A04                lxi d,commonfcb
020C E1C1C5E5              pop h! pop b! push b! push h
0210 221E00                shld user$info

                        nocpyin:

0213 CDD502                call bank$bdos

0216 D1                    pop d ;restore FCB address
0217 C178                  pop b! mov a,b ;restore fcbtbl byte & function #
0219 E6F8C8                ani 0f8h! rz
021C 212A04EB01            lxi h,commonfch! xchg! lxi b,33
0223 17DA4002              ral! jc copy$fcb$back             ;fcbout test
0227 0E2417DA40            mvi c,36! ral! jc copy$fcb$back   ;pfcbout test
022D 17DA6802              ral! jc cdmacpyout128             ;cdmaout128 test
0231 0E0417DA4B            mvi c,4! ral! jc movef            ;timeout test
0237 17DA5C02              ral! jc cdmacpyout003             ;cdmaout003 test
023B 0E06C34BFF            mvi c,6! jmp movef                ;seriout

                        copy$fcb$back:
0240 3A1D00B7CA            lda copy$cr$only! ora a! jz movef
0247 010E0009EB            lxi b,14! dad b! xchg! dad b
024D 7E12                  mov a,m! stax d
024F 2313                  inx h! inx d
0251 7E12                  mov a,m! stax d
0253 03030309EB            inx b! inx b! inx b! dad b! xchg! dad b
0259 1A77C9                ldax d! mov m,a! ret

                        cdmacpyout003:
025C 2AD8050103            lhld dmaad! lxi b,3! lxi d,common$dma
0265 C34BFF                jmp movef

                        cdmacpyout128:
0268 2AD8050180            lhld dmaad! lxi b,128! lxi d,common$dma
0271 C34BFF                jmp movef

                    parse:
0274 EB5E2356              xchg! mov e,m! inx h! mov d,m
0278 234E2346              inx h! mov c,m! inx h! mov b,m
027C 21AF04E5C5            lxi h,buffer+133! push h! push b! push d
0282 222C04212E            shld buffer+2! lxi h,buffer+4! shld buffer
028B 018000CD4B            lxi b,128! call movef! mvi m,0
0293 0E98112A04            mvi c,152! lxi d,buffer! call bank$bdos
029B C17DB4CAB4            pop b! mov a,l! ora h! jz parse1
02A1 7DA43CCAB4            mov a,l! ana h! inr a! jz parse1
02A7 112E04                lxi d,buffer+4
02AA 7D936F                mov a,l! sub e! mov l,a
02AD 7C9A67                mov a,h! sbb d! mov h,a
02B0 09222804              dad b! shld aret
                    parse1:
02B4 E1D1012400            pop h! pop d! lxi b,36! jmp movef
```

```
                    bad$func:
02BC FE98CA7402              cpi 152! jz parse

                             ; A = 0 if fx >= 128, 0ffh otherwise
02C1 173E00DAC8              ral! mvi a,0! jc sta$ret

02C7 3D                      dcr a

                    sta$ret:
02C8 322804                  sta aret

                    goback:
02CB 2A6605F9                lhld entsp! sphl ;user stack restored
02CF 2A28047D44              lhld aret! mov a,l! mov b,h ;BA = HL = aret
02D4 C9                      ret

                    BANK$BDOS:

02D5 AFCD51FF                xra a! call selmemf

02D9 CD06FC                  call bnkbdos

02DC 222804                  shld aret
02DF 3E01C351FF              mvi a,1! jmp selmemf ;ret


                    move$out:
02E4 B7CA4BFF                ora a! jz move$f
02E8 CD51FF                  call selmemf
                    move$ret:
02EB CD4BFF                  call movef
02EE AFC351FF                xra a! jmp selmemf

                    move$tpa:
02F2 3E01CD51FF              mvi a,1! call selmemf
02F7 C3EB02                  jmp move$ret

                    search$hash: ; A = bank # , HL = hash table addr

                             ; Hash format
                             ; xxsuuuuu xxxxxxxx xxxxxxxx ssssssss
                             ; x = hash code of fcb name field
                             ; u = low 5 bits of fcb user field
                             ;     1st bit is on for XFCB's
                             ; s = shiftr(mod || ext,extshf)

02FA 222504CD51              shld hash$tbla! call selmemf
                             ; Push return address
0300 21D603E5                lxi h,search$h7! push h
                             ; Reset read directory record flag
0304 AF321400                xra a! sta rd$dir

0308 2A2504444D              lhld hash$tbla! mov b,h! mov c,l
030D 2A1200EB                lhld hashmx! xchg
                             ; Return with Z flag set if dcnt = hash$mx
0311 2AE105E5CD              lhld dcnt! push h! call subdh! pop d! ora l! rz
```

```
                          ; Push hash$mx-dcnt (# of hash$tbl entries to search)
                          ; Push dcnt+1
031B E513EBE5             push h! inx d! xchg! push h
                          ; Compute .hash$tbl(dcnt-1)
031F 2B292909             dcx h! dad h! dad h! dad b
                   search$h1:
                          ; Advance hl to address of next hash$tbl entry
0323 1104001911           lxi d,4! dad d! lxi d,hash
                          ; Do hash u fields match?
032A 1AAEE61FC2           ldax d! xra m! ani 1fh! jnz search$h3 ; no
                          ; Do hash's match?
0331 CDC003CA7E           call search$h6! jz search$h4 ; yes
                   search$h2:
0337 EBE1                 xchg! pop h
                   search$h25:
                          ; de = .hash$tbl(dcnt), hl = dcnt
                          ; dcnt = dcnt + 1
0339 23E3                 inx h! xthl
                          ; hl = # of hash$tbl entries to search
                          ; decrement & test for zero
                          ; Restore stack & hl to .hash$tbl(dcnt)
033B 2B7DB4E3E5           dcx h! mov a,l! ora h! xthl! push h
                          ; Are we done?
0340 EBC22303             xchg! jnz search$h1 ; no - Keep searching
                          ; Search unsuccessful - return with Z flag reset
0344 3CE1E1C9             inr a! pop h! pop h! ret
                   search$h3:
                          ; Does xdcnt+1 = 0ffh?
0348 3A18003CCA           lda xdcnt+1! inr a! jz search$h5 ; yes
                          ; Does xdcnt+1 = 0feh?
034F 3CC23703             inr a! jnz search$h2 ; no - continue searching
                          ; Do hash's match?
0353 D5CDC003D1           push d! call search$h6! pop d! jnz search$h2 ; no
                          ; Does find$xfcb = 0ffh?
035B 3A16003CCA           lda find$xfcb! inr a! jz search$h45 ; yes
                          ; Does find$xfcb = 0feh?
0362 3CCA6F03             inr a! jz search$h35 ; yes
                          ; xdcnt+1 = 0feh & find$xfcb < 0feh
                          ; Open user 0 search
                          ; Does hash u field = 0?
0366 7EE61FC237           mov a,m! ani 1fh! jnz search$h2 ; no
                          ; Search successful
036C C37E03               jmp search$h4
                   search$h35:
                          ; xdcnt+1 = 0feh & find$xfcb = 0feh
                          ; Delete search to return matching fcb's & xfcbs
                          ; Do hash user fields match?
036F 1AAEE60FC2           ldax d! xra m! ani 0fh! jnz search$h2 ; no
                          ; Exclude empty fcbs, sfcbs, and dir lbls
0376 7EE630FE30           mov a,m! ani 30h! cpi 30h! jz search$h2
                   search$h4:
                          ; successful search
                          ; Set dcnt to search$hash dcnt-1
                          ; dcnt gets incremented by read$dir
                          ; Also discard search$hash loop count
037E 2AE105EB             lhld dcnt! xchg
```

```
 0382 E12B22E100        pop h! dcx h! shld dcnt! pop b
                        ; Does dcnt&3 = 3?
 0388 7DE603FE03        mov a,l! ani 03h! cpi 03h! rz ; yes
                        ; Does old dcnt & new dcnt reside in same sector?
 038E 7BE6FC5F          mov a,e! ani 0fch! mov e,a
 0392 7DE6FC6F          mov a,l! ani 0fch! mov l,a
 0396 CDDD03B5C8        call subdh! ora l! rz ; yes
                        ; Set directory read flag
 039B 3EFF321400        mvi a,0ffh! sta rd$dir
 03A0 AFC9              xra a! ret
                search$h45:
                        ; xdcnt+1 = 0feh, find$xfcb = 0ffh
                        ; Rename search to save dcnt of xfcb in xdcnt
                        ; Is hash entry an xfcb?
 03A2 7EE610CA37        mov a,m! ani 10h! jz search$h2 ; no
                        ; Do hash user fields agree?
 03A8 1AAEE60FC2        ldax d! xra m! ani 0fh! jnz search$h2 ; no
                        ; set xdcnt
 03AF C3B803            jmp search$h55
                search$h5:
                        ; xdcnt+1 = 0ffh
                        ; Make search to save dcnt of empty fcb
                        ; is hash$tbl entry empty?
 03B2 7EFEF5C237        mov a,m! cpi 0f5h! jnz search$h2 ; no
                search$h55:
                        ; xdcnt = dcnt
 03B8 EBE1221700        xchg! pop h! shld xdcnt! jmp search$h25
                search$h6:
                        ; hash compare routine
                        ; Is hashl = 0?
 03C0 3A9C05B7C8        lda hashl! ora a! rz ; yes - hash compare successful
                        ; hash$mask = 0e0h if hashl = 3
                        ;           = 0c0h if hashl = 2
 03C5 4F0F0F1F47        mov c,a! rrc! rrc! rar! mov b,a
                        ; hash s field does not pertain if hashl ^= 3
                        ; Does hash(0) fields match?
 03CA 1AAEA0C0          ldax d! xra m! ana b! rnz ; no
                        ; Compare remainder of hash fields for hashl bytes
 03CE E52313CDE4        push h! inx h! inx d! call compare
 03D4 E1C9              pop h! ret
                search$h7:
                        ; Return to bnkbdos
 03D6 F5AFCD51FF        push a! xra a! call selmemf! pop a! ret

                subdh:
                        ;compute HL = DE - HL
 03DD 7B956F            mov a,e! sub l! mov l,a
 03E0 7A9C67            mov a,d! sbb h! mov h,a
 03E3 C9                ret

                compare:
 03E4 1ABEC0            ldax d! cmp m! rnz
 03E7 23130DC8          inx h! inx d! dcr c! rz
 03EB C3E403            jmp compare

                ;       Disk Function Copy Table
```

```

0001 =        cdmain equ     00000001B        ;copy 1ST 16 bytes of DMA to
                                              ;common$dma on entry
0002 =        fcbin  equ     00000010b        ;fcb copy on entry
0080 =        fcbout equ     10000000b        ;fcb copy on exit
0040 =        pfcbout        equ     01000000b        ;random fcb copy on exit
0020 =        cdma128        equ     00100000b        ;copy 1st 128 bytes of common$dma
                                              ;to DMA on exit
0010 =        timeout equ    00010000b        ;copy date & time on exit
0008 =        cdma003 equ    00001000B        ;copy 1ST 3 bytes of common$dma
                                              ;to DMA on exit
0004 =        serout  equ    00000100b        ;copy serial # on exit

              dfctbl:
03EE 00               db 0                    ; 12=return version #
03EF 00               db 0                    ; 13=reset disk system
03F0 00               db 0                    ; 14=select disk
03F1 83               db fcbin+fcbout+cdmain  ; 15=open file
03F2 82               db fcbin+fcbout         ; 16=close file
03F3 22               db fcbin+cdma128        ; 17=search first
03F4 22               db fcbin+cdma128        ; 18=search next
03F5 03               db fcbin+cdmain         ; 19=delete file
03F6 82               db fcbin+fcbout         ; 20=read sequential
03F7 82               db fcbin+fcbout         ; 21=write sequential
03F8 83               db fcbin+fcbout+cdmain  ; 22=make file
03F9 03               db fcbin+cdmain         ; 23=rename file
03FA 00               db 0                    ; 24=return login vector
03FB 00               db 0                    ; 25=return current disk
03FC 00               db 0                    ; 26=set DMA address
03FD 00               db 0                    ; 27=get alloc address
03FE 00               db 0                    ; 28=write protect disk
03FF 00               db 0                    ; 29=get R/O vector
0400 83               db fcbin+fcbout+cdmain  ; 30=set file attributes
0401 00               db 0                    ; 31=get disk param addr
0402 00               db 0                    ; 32=get/set user code
0403 82               db fcbin+fcbout         ; 33=read random
0404 82               db fcbin+fcbout         ; 34=write random
0405 42               db fcbin+pfcbout        ; 35=compute file size
0406 42               db fcbin+pfcbout        ; 36=set random record
0407 00               db 0                    ; 37=drive reset
0408 00               db 0                    ; 38=access drive
0409 00               db 0                    ; 39=free drive
040A 82               db fcbin+fcbout         ; 40=write random w/ zero fill

040B 82               db fcbin+fcbout         ; 41=test & write record
040C 00               db 0                    ; 42=record lock
040D 00               db 0                    ; 43=record unlock
040E 00               db 0                    ; 44=set multi-sector count
040F 00               db 0                    ; 45=set BDOS error mode
0410 08               db cdma003              ; 46=get disk free space
0411 00               db 0                    ; 47=chain to program
0412 00               db 0                    ; 48=flush buffers
0413 02               db fcbin                ; 49=Get/Set system control block
0414 02               db fcbin                ; 50=direct BIOS call (CP/M)
0033 =        ndf equ ($-dfctbl)+12

```

```
                      xdfctbl:
0415 00                        db 0                    ; 98=reset allocation vectors
0416 03                        db fcbin+cdmain         ; 99=truncate file
0417 03                        db fcbin+cdmain         ; 100=set directory label
0418 00                        db 0                    ; 101=return directory label data
0419 83                        db fcbin+fcbout+cdmain  ; 102=read file xfcb
041A 03                        db fcbin+cdmain         ; 103=write or update file xfcb
041B 02                        db fcbin                ; 104=set current date and time
041C 12                        db fcbin+timeout        ; 105=get current date and time
041D 02                        db fcbin                ; 106=set default password
041E 06                        db fcbin+serout         ; 107=return serial number
041F 00                        db 0                    ; 108=get/set program return code
0420 00                        db 0                    ; 109=get/set console mode
0421 00                        db 0                    ; 110=get/set output delimiter
0422 00                        db 0                    ; 111=print block
0423 00                        db 0                    ; 112=list block

0071 =                nxdf equ ($-xdfctbl)+98

0424                  res$fx:        ds      1
                      hash$tbla:
0425                           ds      2
0427                  bank:    ds      1
0428                  aret:    ds      2       ;address value to return

                      buffer:                          ;function 10 256 byte buffer

                      commonfcb:
042A                           ds      36      ;fcb copy in common memory

                      common$dma:
044E                           ds      220     ;function 10 buffer cont.

052A                           ds      ssize*2
                      lstack:
0566                  entsp: ds      2

                      ; BIOS intercept vector

0568 C303FF           wbootfx:       jmp     wbootf
056B C38301                          jmp     switch1
056E C306FF           constfx:       jmp     constf
0571 C38301                          jmp     switch1
0574 C309FF           coninfx:       jmp     coninf
0577 C38301                          jmp     switch1
057A C30CFF           conoutfx:      jmp     conoutf
057D C38301                          jmp     switch1
0580 C30FFF           listfx:                jmp     listf
0583 C38301                          jmp     switch1

0586 0000000000                      dw      0,0,0
058C 00000000                        dw      0,0

0590 0000             olog:          dw      0
0592 0000             rlog:          dw      0

```

```
0594 00000000  patch$flgs:    dw      0,0

               ; Base of RESBDOS

0598 0600             dw      base+6

               ; Reserved for use by non-banked BDOS

059A                  ds      2

               ; System Control Block

               SCB:

               ; Expansion Area - 6 bytes

059C 00        hashl:         db      0       ;hash length (0,2,3)
059D 00000000  hash:          dw      0,0     ;hash entry
05A1 31        version:       db      31h     ;version 3.1

               ; Utilities Section - 8 bytes

05A2 00000000  util$flgs:     dw      0,0
05A6 0000      dspl$flgs:     dw      0
05A8 0000                     dw      0

               ; CLP Section - 4 bytes

05AA 0000      clp$flgs:      dw      0
05AC 0000      clp$errcde:    dw      0

               ; CCP Section - 8 bytes

05AE 00        ccp$comlen:    db      0
05AF 00        ccp$curdrv:    db      0
05B0 00        ccp$curusr:    db      0
05B1 0000      ccp$conbuff:   dw      0
05B3 0000      ccp$flgs:      dw      0
05B5 00                       db      0

               ; Device I/O Section - 32 bytes

05B6 00        conwidth:      db      0
05B7 00        column:                db      0
05B8 00        conpage:       db      0
05B9 00        conline:       db      0
05BA 0000      conbuffadd:    dw      0
05BC 0000      conbufflen:    dw      0
05BE 0000      conin$rflg:    dw      0
05C0 0000      conout$rflg:   dw      0
05C2 0000      auxin$rflg:    dw      0
05C4 0000      auxout$rflg:   dw      0
05C6 0000      lstout$rflg:   dw      0
05C8 00        page$mode:     db      0
05C9 00        pm$default:    db      0
05CA 00        ctlh$act:      db      0
```

```
05CB 00         rubout$act:     db      0
05CC 00         type$ahead:     db      0
05CD 0000       contran:        dw      0
05CF 0000       conmode:        dw      0
05D1 6A04                       dw      buffer+64
05D3 24         outdelim:       db      '$'
05D4 00         listcp:                 db      0
05D5 00         qflag:          db      0

                ; BDOS Section - 42 bytes

05D6 9C05       scbadd:                 dw      scb
05D8 8000       dmaad:          dw      0080h
05DA 00         seldsk:                 db      0
05DB 0000       info:           dw      0
05DD 00         resel:          db      0
05DE 00         relog:                  db      0
05DF 00         fx:             db      0
05E0 00         usrcode:        db      0
05E1 0000       dcnt:           dw      0
05E3 0000       searcha:        dw      0
05E5 00         searchl:        db      0
05E6 01         multcnt:        db      1
05E7 00         errormode:      db      0
05E8 00FFFFFF   searchchain:    db      0,0ffh,0ffh,0ffh
05EC 00         temp$drive:     db      0
05ED 00         errdrv:         db      0
05EE 0000                       dw      0
05F0 00         media$flag:     db      0
05F1 0000                       dw      0
05F3 80         bdos$flags:     db      80h
05F4 FFFFFFFFFFstamp:           db      0ffh,0ffh,0ffh,0ffh,0ffh
05F9 0000       commonbase:     dw      0
05FB C37CFC     error:                  jmp     error$jmp
05FE 0600       bdosadd:        dw      base+6
0600                            end
```

```
ARET               0428   110    333    346    350    359    584#
AUXINRFLG          05C2   667#
AUXINSTF           FF36    61#   160
AUXOUTRFLG         05C4   668#
AUXOUTSTF          FF39    62#   163
BADFUNC            02BC   234    235    337#
BANK               0427   583#
BANKBDOS           02D5   126    126    128    184    209    291    327    353#
BASE               0000    32#    34     35     36     37    621    707
BDOS               0024    79    105#
BDOSADD            05FE   707#
BDOSFLAGS          05F3   703#
BIOS               FF00    42#
BIOSPG             FF00    37#    42     43     44     45     46     47     48     49     50
                           51     52     53     54     55     56     57     58     59     60
                           61     62     63     64     65     66     67     68     69     70
                           71     72
BLKOUT             0141   196#   213
BLKOUT0            0153   171    202#
BLKOUT1            016A   203    204    208#
BLKOUT2            0170   199    211#
BNKBDOS            FC06    39#   357
BNKBDOSPG          FC00    34#    39     40
BOOTF              FF00    43#
BUFFER             042A   177    180    185    324    325    325    325    327    330    586#
                          677
CCPCOMLEN          05AE   650#
CCPCONBUFF         05B1   653#
CCPCURDRV          05AF   651#
CCPCURUSR          05B0   652#
CCPFLGS            05B3   654#
CDMA003            0008   514#   554
CDMA128            0020   511#   524    525
CDMACPYOUT003      025C   301    313#
CDMACPYOUT128      0268   299    317#
CDMAIN             0001   506#   522    526    529    530    537    563    564    566    567
CHAINFX            002F    28#
CLPERRCDE          05AC   646#
CLPFLGS            05AA   645#
COLUMN             05B7   660#
COMMONBASE         05F9   705#
COMMONDMA          044E    98    277    314    318    591#
COMMONFCB          042A   283    285    296    588#
COMPARE            03E4   487    499#   502
CONBUFFADD         05BA   177    663#
CONBUFFLEN         05BC   664#
CONIN              00D8   149    155#
CONINF             FF09    46#   157    604
CONINFX            0574   604#
CONINRFLG          05BE   665#
CONLINE            05B9   662#
CONMODE            05CF   203    676#
CONOUTF            FF0C    47#    74    143    206    606
CONOUTFX           057A   606#
CONOUTFXX          0002    24#   209
CONOUTRFLG         05C0   666#
```

```
CONOUTSTF       FF33    60#
CONPAGE         05B8   661#
CONSTATFX       000B    26#
CONSTF          FF06    45#  153   602
CONSTFX         056E   602#
CONSTX          00CE   145   147   151#
CONTRAN         05CD   675#
CONWIDTH        05B6   659#
COPYCRONLY      001D   100#  282   305
COPYFCBBACK     0240   297   298   304#
CPYCDMAIN       01ED   271   275#
CPYFCBIN        01FC   272   281#
CTLHACT         05CA   672#
CURDMA          001B    99#  246
DCNT            05E1   392   448   449   692#
DEVINITF        FF3F    64#
DEVTBLF         FF3C    63#
DFCTBL          03EE   261   518#  559
DIRINP          00C3   140   146#
DIRINP1         00C8   142   148#
DIRSTAT         00BD   141   144#
DISKFUNC        0194   114   232#
DISKFUNCTION    01CF   238   243   254#
DISKFX          000C    23#  114
DMAAD           05D8   178   246   276   314   318   685#
DRVTBLF         FF42    65#
DSPLFLGS        05A6   640#
ENTSP           0566   111   349   596#
ERRDRV          05ED   699#
ERROR           05FB   706#
ERRORJMP        FC7C    40#  706
ERRORMODE       05E7   696#
FCBIN           0002   508#  522   523   524   525   526   527   528   529   530
                       537   540   541   542   543   547   549   557   558   563
                       564   566   567   568   569   570   571
FCBOUT          0080   509#  522   523   527   528   529   537   540   541   547
                       549   566
FINDXFCB        0016    95#  427
FLUSHF          FF48    67#
FUNC10          0101   128   174#
FUNC10A         0110   176   179#
FUNC111         0135   189#  236
FUNC112         0135   190#  237
FUNC3           009E   126   130#
FUNC4           00A4   127   133#
FUNC5           00A8   127   136#
FUNC6           00AC   127   139#
FUNC7           00E3   127   159#
FUNC8           00E9   128   162#
FUNC9           00EF   128   165#
FUNCTAB         0086   116   125#
FX              05DF   690#
GOBACK          02CB   112   348#
HASH            059D   400   634#
HASHL           059C   479   633#
HASHMX          0012    92#  390
```

```
HASHTBLA        0425   383    389    581#
HOMEF           FF18    51#
INFO            05DB   687#
IOLOC           0003    29#
KBCHAR          0020   102#   152    156
LISTCP          05D4   679#
LISTF           FF0F    48#   137    200    608
LISTFX          0580   608#
LISTSTF         FF2D    58#
LSTACK          0566   112    595#
LSTOUTRFLG      05C6   669#
MAKEXFCB        0015    94#
MEDIAFLAG       05F0   701#
MOVEF           FF4B    68#   182    187    278    284    300    302    305    315    319
                       326    335    364    367
MOVEOUT         02E4    80    363#
MOVERET         02EB   366#   372
MOVETPA         02F2    82    370#
MULTCNT         05E6   695#
MULTIOF         FF45    66#
NDF             0033   233    559#
NOCPYIN         0213   273    289#
NORMALCPM       01DB   262    264#
NXDF            0071   235    578#
OKDF            01B0   233    240#
OLOG            0590   614#
OUTDELIM        05D3   168    678#
PAGEMODE        05C8   670#
PARSE           0274   321#   338
PARSE1          02B4   328    329    334#
PATCHFLGS       0594   617#
PFCBOUT         0040   510#   542    543
PMDEFAULT       05C9   671#
PRINT           00F1   167#   172
PRINTFX         0009    25#
PUNCHF          FF12    49#   134
QCONINX         0177   103    215#
QFLAG           05D5   680#
RDDIR           0014    93#   387    457
READERF         FF15    50#   131
READF           FF27    56#
RELOG           05DE   689#
RESBDOSPG       FD00    35#
RESEL           05DD   688#
RESFX           0424   191    200    580#
RLOG            0592   615#
RUBOUTACT       05CB   673#
SCB             059C   629#   684
SCBAL           05D6   684#
SCBPG           FE00    36#
SCONOUTF        FF0C    74#   205
SCREENF         FFFF    75#
SEARCH          01C7   241    248#
SEARCHA         05E3   249    252    693#
SEARCHCHAIN     05E8   697#
SEARCHH1        0323   398#   416
```

```
SEARCHH2       0337   405#  423   425   433   440   442   463   465   472
SEARCHH25      0339   407#  475
SEARCHH3       0348   402   419#
SEARCHH35      036F   429   436#
SEARCHH4       037E   404   435   443#
SEARCHH45      03A2   427   459#
SEARCHH5       03B2   421   468#
SEARCHH55      03B8   467   473#
SEARCHH6       03C0   404   425   476#
SEARCHH7       03D6   385   489#
SEARCHHASH     02FA    84   374#
SEARCHL        05E5   694#
SEARCHN        01CB   242   251#
SECTRAN        FF30    59#
SELDSK         05DA   686#
SELDSKF        FF1B    52#
SELMEMF        FF51    70#  217   221   227   229   355   360   365   368   371
                      383   491
SERIAL         0000    77#
SEROUT         0004   516#  571
SETBNKF        FF54    71#
SETDMAF        FF24    55#
SETDMAFX       001A    27#  243
SETSECF        FF21    54#
SETTRKF        FF1E    53#
SSIZE          001E    22#  594
STAMP          05F4   704#
STARET         02C8   131   145   149   160   163   341   345#
SUBDH          03DD   392   455   493#
SWITCH0        018D   226   228#
SWITCH1        0183   225#  601   603   605   607   609
TEMPDRIVE      05EC   698#
TIMEF          FF4E    69#
TIMEOUT        0010   513#  569
TYPEAHEAD      05CC   674#
USERINFO       001E   101#  287
USRCODE        05E0   691#
UTILFLGS       05A2   639#
VERSION        05A1   635#
WBOOTF         FF03    44#  126   600
WBOOTFX        0568   600#
WRITEF         FF2A    57#
XDCNT          0017    96#  421   475
XDFCTBL        0415   263   561#  578
XDMAADD        0019    98#
XMOVEF         FF4E    72#
```

CP/M - PLUS V3.0

P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # CP3 -135